Norwegian School of Economics

Bergen, Fall, 2019

Automatic Machine Learning Applied to Time Series Forecasting for Novice Users in Small to Medium-Sized Businesses

A review of how companies accumulate and use data along with an interface for data preparation as well as easy and powerful prediction analysis capable of providing valuable insight

Anders Stykket Gran

Supervisor: Håkon Otneim

Master thesis, Economic Analysis

NORWEGIAN SCHOOL OF ECONOMICS

Abstract

Data analytics is gradually becoming one of the most essential tools and sources of competitive advantage for modern companies. There is a multitude of analytical services and solutions on the market, and the effect of data analytics is both well documented and significant. One of the more underutilized aspects of data analytics, especially for small to medium-size businesses, is the making of time series predictions or forecasts based on all available and relevant data. Such companies do not often have their own data scientists and limited resources to invest in learning or developing data analytical competence (Henke et al., 2016). However, there have been great developments in the field of automatic machine learning, making it much easier to create high-quality models without the need for expertly customizing a model to the data. In this thesis, I develop an interface for both data preparation and automatic machine learning that lets novice users apply the full power of H2O AutoML for easy data analytical insight into the unobserved future.

Contents

1. Introduction

Ever since starting at NHH, I have heard lectures and read articles claiming that most businesses are critically underutilizing their data and that more management decisions could and should be based on analysis of data. According to McKinsey, modern society had only captured about 25% of the potential value from data analytics in 2016, as identified in 2011 (Henke et al., 2016). Moreover, this potential value is in the hundreds of billions of dollars. All of this sounds somewhat fantastical, so I wanted to explore how essential data analytics truly is, and whether there were actual realized examples to support such claims. Further, I wanted to study the available data analytical tools and services and then make my contribution by designing an application in the R programming language.

The aspect of data analytics that I find most interesting and potentially useful for management decisions is prediction. Therefore, and because I do not have nearly enough capacity to create a more comprehensive tool, I built an interface for the AutoML algorithm, which is created and maintained by H2O.ai as an open-source automatic machine learning solution. My application, or interface, is aimed at small to medium-sized businesses that are interested in predicting future values but do not have the resources to buy a more complete professional solution. Because preparing data for analysis is such a critical part, and possibly underestimated by those with limited experience, I have included functions designed to allow even a novice user to upload their datasets and prepare them as part of the analysis. With minimal user input, large datasets can be passed to AutoML, and with sufficient computation time and power, AutoML can deliver accurate predictions.

Ease of use and ease of understanding has been a critical concern of mine, which is why I have used the **shiny** package for R to create an application rather than having users interact with a more traditional programming interface (Chang et al., 2019). The application could be made to run on a server, but that is an untapped potential for now, as I have designed everything to run on a local computer. My research question is therefore:

> Can data preparation and automatic machine learning capabilities be rendered to users in such a way that even those without any experience with predictive analytics could extract accurate and useful predictions, only using freely available tools?

The structure of the thesis follows my initial interests closely. Chapter 2 defines data analytics and presents evidence for why it is so important and why companies should be using more of it. Chapter 3 presents and discusses some prominent tools and services for data analysis. Chapter 4 serves as a quick introduction in data preparation and H2O AutoML, along with a short evaluation of the quality of AutoML. Chapter 5 presents the actual interface, how it works and why I chose the presented solutions. Chapter 6 compares my application to two professionally made alternatives based automatic machine learning algorithms. Finally, chapter 7 concludes by evaluating my application and pointing out the potential future for my application.

2. Data analytics in the private sector

There can be little doubt that the amount of data available to decision makers and the quality of the tools to analyze them have grown at an incredible speed in recent years. The availability of data has improved primarily because of two factors, the cost of storing data has fallen, and the way we interact has shifted to mediums that generate much more useful data[1]. Alongside this, computers have gotten exponentially more powerful and new software, like machine learning, allow the computers to automate some of the analytical processes. However, just what do we mean by data analytics?

2.1 Definition of data analytics

According to Investopedia: "Data analytics is the science of analyzing raw data in order to make conclusions about that information," which is a straightforward and general definition, not dependent on the utilized techniques or purpose of the analysis (Frankenfield, 2019). Without the specification of it being a science, and depending on the interpretation of raw data, analysis of raw data to make conclusions is something all people do all the time and thus not very interesting to discuss in this paper. Therefore, what is it that makes data analytics a science, and what is a natural definition or restriction of raw data in this setting?

When referring to data in this paper, I will be using the second definition presented in the Merriam-Webster dictionary: "information in digital form that can be transmitted or processed." The first definition: "factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation" includes the requirement for data to be factual, something one should undoubtedly hope for, but errors, such as measurement errors, can creep into real-world datasets. The third definition: "information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful," brings us back to allowing everything to be considered as data. (Mirriam-Webster, 2019)

[1] In a way, the average modern business-consumer interaction generates much less data, because in an interaction with the actual consumer in the flesh, one could theoretically record all the mannerisms and biometrics. While an online interaction only generates data based on what is transmitted by the consumer's device.

The “raw” part of raw data means data that has not been processed in any way other than the act of its recording and transmission. While it is not the subject of this paper, let me stress that the recording of data is a critical process, something that will often simplify the source material and thus all data analysts should be aware of how we record data and how our data was recorded.

To see what makes data analytics a science and not just any data processing procedure, one can look to one of the first papers that defined data science “The Future of Data Analysis,” written by John W. Tukey in 1962. A traditional definition of science is 1. There must be intellectual content; 2. It must be organized into an understandable form, and 3. It relies upon the test of experience as the ultimate standard of validity. The third one is by far the most complicated, and it excludes mathematics from being defined as a science. In more colloquial terms, data analytics must rely on proven or provable methods, and the results must be reproducible so that they can be tested.

So, by expanding on the initial definition of data analytics, I will use the following definition in this thesis:

> Data analytics is the science of analyzing raw digital information, using proven techniques to make conclusions about that information, which can be reproduced and tested.

2.2 The importance of data analytics

Data analytics comes in many forms, including business intelligence, predictive analytics, “big data,” and machine learning[2]. Some of those terms are more popular than others, but they are all variants of data analytics. The applications are, naturally, almost innumerable, and many such applications should raise serious ethical concerns. I will focus on the private sector and avoid ethical discussions in this paper, which is an important distinction because I assume the goal of almost all private businesses is or should be to maximize the long-term profits of the

[2] Machine learning as a term could also be used to describe training robots to walk, programs to play games etc. In this setting I only mean the usage of programs on datasets with the aim of drawing conclusions from that data.

owners and the profit motive gives an excellent way to quantify the effect of data analytics. Let me digress a little to defend that argument. I emphasize that I mean long-term profits because I am sure many would argue that the increasing focus on Corporate Social Responsibility (CSR) is a departure from a profit-maximizing goal.

When advocating CSR, promoters often mention that those who do well in CSR are more successful than those who focus on maximizing profits, which is, in fact, a profit-maximizing argument for CSR (Murphy, 2018). The problem for those who get called out as unlikable profit maximizers could instead be that they have focused too much on short term profits, been unrealistic in their estimation of the probability of a scandal going public or underestimated the extent of the damage from a scandal. However, it would be counterproductive for a company to publicly state that they only use CSR to maximize profits so one should keep that motivation between management and the board of directors.

At the core, all business to consumer activity is about providing a product or service that the customer more or less subjectively values higher than the cost to provide it and setting the price somewhere in between those points. Therefore, data analytics can help increase long term profits in three main ways; reducing costs, increasing the consumer's valuation of a product/service or more accurately identify each consumer's willingness to pay for the business to capture more of the value creation. Which application that is most relevant is, of course, dependent on the industry, competition, regulations, and other characteristics of the environment in which each company operates. Depending on what data is provided and what variable is predicted, my application can be used in all three ways.

Cost reduction is a common and standard application for data analytics. For example, analyzing employee data to reduce turnover, which cost US employers an estimated 600 billion dollars in 2018 (Fox, 2018), is growing in popularity. The software can identify which employees are likely to leave their current position and who could be persuaded to stay. An estimated 77% of turnover in 2018 was preventable (Fox, 2018). People are applying data analytics to almost every aspect of Human Resources management, with more than 1 400 technology vendors offering all sorts of HR tools in 2019. (Volini et al., 2019)

In a case study performed by Hopkins and Hawking in 2017, they found that outfitting the fleet of a large logistics company with sensors and analyzing the data provided by those sensors allowed the company to cut CO2 emissions and improve driver safety. The software

could recognize signs of driver fatigue and the system warned drivers of upcoming traffic hazards, in addition to serving as the basis for a new training program. Similarly, UPS analyzed and optimized their delivery services and was able to reduce fuel consumption by 147,6 million liters and distance traveled by 585.8 million kilometers (Dryden Group, 2019).

Managing indirect costs is possibly the area where data analytics can have the most significant impact, as indirect expenses account for between 15 and 27% of the revenue for the average company. Indirect costs include travel, utilities, office supplies, and professional services and are often difficult to record, let alone manage. Modern information systems are capable of quantifying almost all of this, and the sheer volume generated by such systems often classifies this as a big data problem. Because most of the personal expenses can be quite minute, they are often left to the discretion of the individual employees who utilize them, and it can be tough to budget for and control such costs. However, by thorough analysis, one can uncover spending patterns across the organization, allowing such decisions to be optimized and centralized. (Dryden Group, 2019)

Advanced data analytics can speed up product testing by processing results much more efficiently as well as providing more profound insights, reducing implementation times and thus costs. Chime Bank used an artificial intelligence platform to assist in the development of a new webpage, allowing the company to try 21 different ideas and 216 different versions of the webpage over three months, something that would have taken nine years with the more straightforward A/B tools they usually used. (Yalif, 2018)

Cybersecurity is another highly relevant application of data analytics, with the average cost of a cyber attack up to 1.1 million in 2019, of which operational/productivity loss accounts for 54% and 43% is negative customer experiences. By monitoring and analyzing internet traffic in real-time, systems can uncover suspicious activity and block malicious devices. Such monitoring is becoming increasingly important, as the rise of the Internet of Things have drastically expanded the number of devices connected to the internet, many of which have little to no cybersecurity installed. Hackers can, therefore, for example, relatively quickly appropriate thousands of such devices and use them to perform one of the most common cyber attacks, the denial of service attack, in which hackers bombard a website with access requests to such an extent that it crashes and becomes unusable to the actual customers. Analytical tools, often using AI, can identify such a surge and use data on connecting devices to prevent server shutdown and still allow customers to continue as usual. (Radware, 2019)

An application that is in the realm of both cost reduction and value generation is in the field of customer services. NewVoiceMedia reports that poor customer service cost companies more than 75 billion dollars per year in 2018, 13 billion more than in their 2016 report. An essential counterpoint to the argument for more analytics in this field is that 68% of surveyed customers said that they wanted an emotional connection with a customer service agent. Still, data analytics can assist in even this aspect, for example through real-time analysis of the customers' tone of voice and word choice on their initial contact to determine which customers require this emotional connection and who will be satisfied with more automated options. (Hyken, 2018)

Data analytics can generate value through improving the customer experience with a given product or service, but also uncover changes to existing products/services or even entirely new products/services that customers would appreciate more than what is available today. Analytical systems can also enable new types of services and customer interactions that were not possible before. An important distinction here is the separation of just using IT systems and data analytics, because some data needs to undergo real analysis, regardless of the level of automation.

A relatively famous example is how the retailer Target managed to predict pregnancies by analyzing shopping patterns, to send targeted advertising for related products. Their campaign was not just for pregnancies, however. Target hypothesized that customers who were going through life-changing events, such as pregnancy, marriage, childbirth, and divorce were more susceptible to changing their shopping habits and therefore, prime targets for advertising. To use terminology closer to my definitions; such customers were receptive to having their valuation of products adjusted. Target was most successful with the famous pregnancy predictions and managed to significantly increase the sales of related products after the launch of their campaign. (Duhigg, 2012)

Targeted advertising seems to be everywhere these days, made especially apparent by the implementation of the General Data Protection Regulation (GDPR). Facebook, the 6th highest valued publicly traded stock company in the world, bases almost their entire business model on running targeted advertisement (Picardo, 2019). And speaking of Facebook, social media analysis is another popular application of data analytics. By combining data on the number of views, likes/dislikes, sharing and the demographical data of the users submitting those metrics across all available social media platforms, companies can evaluate their online impact very

thoroughly. Such analysis has also evolved into evaluating the online impact of individuals, with companies such as PeerIndex and Klout rating the online influence of users. Thereby allowing companies to co-opt this influence by "working" with people with high influence scores or "influencers," often giving them free products or services with the understanding that they should promote them, if not paying them outright. (Schaefer, 2012)

Another application is the use of data analytics to provide decision support, sometimes referred to as decision science. One technique is to crowdsource extensively in the decision process, setting up systems to receive feedback and possibly even ideas from the public. What makes this analytical is the inclusion of systems to determine the value, feasibility, validity, and fit of submitted ideas and reviews. Further, scientists can use text and sentiment analysis, on submitted ideas as a continuation of the previous technique, but also on external forums and other social media to evaluate how the general public receives a promoted product and use this to make adjustments before the final release. Whirlpool, a home appliance manufacturer, had success with this back in 2009. They used Attensity360 to monitor all public conversations related to their brand and integrated it into their models to predict customer churn[3], loyalty, and satisfaction. By devoting resource to understand the feedback, Whirlpool improved their understanding of their own business and improved overall customer satisfaction and service responsiveness. (IDC, 2011)

Decision support is the one of the more interesting applications, and as McKinsey put it in their report "The age of analytics: Competing in a data-driven world": "Above all, data and analytics can enable faster and more evidence-based decision making." They stress the importance of machine learning in particular. Their previous research has estimated that 45% of all work activity can be automated, and they believe machine learning accounts for 80% of that. The even more powerful and much more computation intensive deep learning might push automation beyond 45%. Depending on how things go with projects like Google's Deep Mind

[3] Churn is the percentage of customers of a specific company that stopped using their products/services in a given period.

and Google Brain or Open AI,[4] we might even see synthetic humanoid workers, despite the lack of actual strong/general AI[5]. (Henke et al., 2016)

The report defines four areas of high impact: radical personalization, predictive analytics, strategic optimization, and real-time optimization in operations and logistics. Using advanced demand forecasting and supply planning, one stamping parts producer managed to reduce production costs by 15%. A media company used machine learning to forecast churn and managed to accurately identify 20% of the customers that would stop using their services. A UK bank managed to predict fraudulent transactions with 90% accuracy, using machine learning. By identifying clients with higher accident rates, predictive analytics improved profitability by more than 10% for an insurance company. Despite this, the implementation of such techniques has been relatively limited so far. In fact, recent research has found that investing in data analytics capabilities give exceptionally high returns, with the average business realizing 6 to 8% in productivity gains. That entails doubling the investment within a decade, which is a better rate of return than investments in computer technology in the 1980s. (Henke et al., 2016)

It seems quite clear that data analytics has significant potential for generating value and that there are many applications, including what appears to be appropriate uses for my application. However, other than the mentioned examples, how is the private sector implementing data analytics?

2.3 Implementation of data analytics in the private sector

In 2011, McKinsey made estimates of the potential value of data analytics in different industries, and in 2016 they revised those estimates and evaluated how much value that had been captured. The best performance was with location-based services, were 50-60% of the 100-billion-dollar increase in revenue had been captured, with the rest being held back by the

[4] A think tank set up primarily by Elon Musk and Sam Altman.

[5] AI or artificial intelligence is a wide field, encompassing both machine learning, deep learning and many other applications. Strong AI or general AI refers to AI so advanced that it can do everything humans can and more, which is the subject of much science fiction.

penetration of GPS-enabled smartphones in the global market. The US and EU retail market has captured about 30 to 40% of the estimated 60% increase in net margin, but are being held back by lack of analytical talent and the fact that each company keeps their recorded data to themselves. Manufacturing has only captured about 20 to 30% of the estimated benefits, those being up to 50% lower product development costs, up to 25% lower operating costs and an up to 30% increase in gross margin. The manufacturing sector is struggling due to much of their data being stored in old IT systems, and much of the leadership still being skeptical of the impact of more data analytics. (Henke et al., 2016)

The US health care system has gotten the least of their value estimate, only capturing 10 to 20% of the potential 300 billion dollars and 0.7% annual productivity growth. The problem for health care is the need to demonstrate clinical utility to gain acceptance and difficulties due to the sensitive nature of their data. To summarize, the private sector had implemented about 25% of the potential utility from data analytics in 2016, with massive differences between the leading firms and the average firm as well as across industries. (Henke et al., 2016)

In 2019 NewVantage Partners released a survey of senior corporate executives on the topics of Big Data and AI, including nearly 65 Fortune 1000 or industry leading participants. 91.6% report that the pace of investment in Big Data and AI is increasing. The key findings are in table Table 2-1.

Percentages show the amount of positive responses	2017	2018	2019
Is your company data-driven?	37.1%	32.4%	31.0%
Is your company managing data as an asset?			46.9%
Does your company have a data culture?			28.3%
Is your company driving innovation with data?			59.5%
Is your company competing on data and analytics?			47.6%
Has your company gotten measurable results on data?			62.2%

Table 2-1 NewVantage Partners Big Data and AI 2019 survey (Bean, 2019)

Interrestingly, the number of firms who claim to be data-driven has decreased over the past three years, and only 28.3% claim to have a data culture in 2019. Therefore, it seems that while companies are investing and innovating with data, they are struggling much more with transforming their core operations to include data. 77.1% of the companies report that the adaptation of Big Data and AI initiatives remains a significant challenge. The reasons for this

challenge were reported as follows: Lack of organizational alignment/agility 40.3%, cultural resistance 23.6%, understanding data as an asset 13.9%, executive leadership 7.0% and technological solutions 5.0%. That means 95% of the problem lies with the people and processes of the average organization. (Bean, 2019)

The companies that have successfully implemented data capabilities are using them to gain substantial advantages. Apple, Alphabet/Google, Amazon, Facebook, Microsoft, GE, Baidu, Alibaba Group and Tencent are some of the most valuable companies in the world, in fact, all but GE and Baidu are in the top ten most valuable companies. What differentiates these companies are their unique data sources, a wealth of analytical talent and their investments in data infrastructure. Besides, several newcomers are revolutionizing their fields with new data analytics-based business models, such as Airbnb, BlaBla Car, Didi Chuxing, DJI, Flipkart, Lyft, Ola, Palantir, Pinterest, Snapchat, Snapdeal, Spotify and Uber. As technology allows more of a market to be digitized, it is not just an opportunity for the companies in that market, it can also present a significant threat, as new companies might ignore traditional entry barriers and completely change the market dynamics. Airbnb, Amazon, Netflix, Uber, and many unique “fintech” companies have done just that. An additional risk with digitization is that the network effects can create a winner-takes-most scenario, where their private access to most of the data can be a significant advantage over new entrants. (Chui & Manyika, 2015)

Established companies do not just have to worry about new entrants either, who might be bought up or managed due to their lack of capital. Several of the tech-giants mentioned above are actively looking for industries to disrupt with their unique capabilities. Alphabet has expanded into autonomous vehicles, Apple into finance with Apple Pay, and Alibaba managed to get better performance on micro-loans than traditional banks, to name a few. This aggressiveness means incumbents need to maintain a two-part strategy regarding data analytics, both looking for high-risk, high-reward opportunities and working to transform their core business to be more data-driven. (Dobbs et al., 2015)

2.4 The way forward for data analytics in the private sector

A fundamental problem for many companies is the lack of analytical talent, as well as cultural issues. In 2016, 50% of executives surveyed by McKinsey reported that they had more trouble

with recruiting analytical talent than for any other role (Brown & Gottlieb, 2016). Further, the salary of data scientists rose by about 16% from 2012 to 2014, compared to overall wage growth of about 2% (Sinclair, 2015).

Many education institutions have responded by expanding their data analytics courses, producing an estimated 7% annual increase in the number of data scientists; however, estimated demand growth is up to 12%. This is good news for data scientists, and probably for consultancies that deal with organizational change as well. However, companies must not forget the need for competence in adapting the data analytics to their organization, knowing what questions to ask and how to get the right insights from analytical results. This translation requires in-depth organizational knowledge and familiarity with each business, and so it would be beneficial for many companies to develop such competence internally. As data analytics offers deeper and deeper insights, understanding, and transmitting those insights becomes harder. Graphical visualization is a handy tool in such endeavors; in fact, the demand for visualization grew by an estimated 50% from 2010 to 2015. Today, individual data scientists are often expected to perform the visualization themselves, but there will likely be a shift to a new type of specialist with both data understanding and graphical skills, focused on creating suitable user interfaces and visualizations. (Henke et al., 2016)

Becoming a data-driven organization is a multi-step problem that requires significant effort on all levels of the organization. McKinsey has a five-step process aimed towards those who are not digital natives. The first step is quite natural and fundamental; one must determine what the data analytics will be used for, what insights are needed, how they will generate value and how one will measure that value. Next is improving the data architecture, making sure as many interactions as possible are digitized, that all data is stored in an accessible format and adding systems for importing external data where necessary. The third element is acquiring the analytical capabilities, as discussed previously. The two final items are where the process becomes challenging, requiring a high degree of customization. The fourth step is to change business processes to include data insights into the workflow. That means making sure insights are displayed in an understandable format to those who can benefit from them throughout the organization; this step also includes the automation of tasks where available. The final step is to educate executives and middle-level managers on how to use the insights as a basis for making decisions. All of this makes it clear that merely buying an analytical tool is not enough. (Henke et al., 2016)

There are a lot of analytical services on the market, many much more advanced than a single tool. Large corporations often acquire analytical talent by buying such service providers or specialist companies. In such deals, the price per employee is usually between five and ten million dollars, such as Google acquiring DeepMind Technologies for nearly seven million dollars per employee (Shu, 2014). Naturally, a small to medium sized business cannot afford to buy a whole data analytics company to internalize needed competence. So what kind of services are available to those looking for external expertise?

3. Analytical tools and services

There is a multitude of programs and services available on the market. For this thesis, I define analytical tools and services as those that enable or perform data analytics as I have specified data analytics. First, I will go over the tools that companies can use on their own, and could be alternatives to my application, and then the services that hopefully give value to their customers without them having to perform analysis themselves. Further, there are too many such services and tools for me to mention all of them, so I will limit myself to a selection of those that have received favorable reviews, for example in one or more of the "top x" lists.

3.1 Analytical tools

What is currently considered as the top analytical tool in the industry is the open-source programming language R, (Sandeep, 2018). R is lauded for having excellent capacity, speed and flexibility. With 14,199 packages as of today, R is full of specialized tools, and with some extensions, it can even be used for Deep Learning. I have used R to create my tool, relying on the Shiny package for the user interface and so I can attest personally to the flexibility and power of R. However, someone with no programming experience could not just download R and expect to analyse all their business problems right away. R requires that the user formulates the entire problem setup and techniques to use, in detail. There are excellent support forums available, such as Stack Overflow, that can allow even novice users to adapt professional solutions to their problems. There are many extensions for R, like the open-source interface RStudio, which make R easier to use. (Sandeep, 2018)

Excel is perhaps the most widely used of all analytical tools. It is not as powerful or flexible as R and many other applications, but with some extra packages, it can handle a lot of analytical problems. Even those who are experts in R or some other programming tool will still probably need to use Excel frequently, for example, to access the internal data of a company or to cooperate with people who are not data scientists. I have also used Excel extensively and for many problems there is no need for advanced tools such as R. One of the main issues with Excel is also one of its strengths, the fact that it shows all the data in cells makes it easy to inspect visually, but it also gives Excel a lot of problems with large datasets. Some of this can be avoided by keeping the data source in a closed workbook and with Visual

Basic script it is possible to perform advanced analysis on excel sheets without really opening them.

Tableau is more focused on visualization and includes advanced functionality for real-time tracking and sharing of data. Users can combine several data sources easily and create many different plots. Tableau can also perform simple forecasting, based on exponential smoothing, which is relatively simple and only based on the history of the variable that is being predicted. Therefore, it is only relatively accurate if what is being forecasted behaves like it has previously. More complicated models, like the one I have created, try to find other forward-looking variables that enable more or less accurate predictions in the absence of strong trends. (Tableu, 2019)

Python is more like R, in that it is much more flexible and open ended, capable of doing almost anything, but it requires programming from the user. It is an object-based scripting language that is considered to be easy to learn, write and maintain and it is open source. Python is particularly good at handling machine learning, and I certainly could have made my appliaction with Python instead. (Sandeep, 2018)

SAS will feature as both an analytical software tool and a service, due to being a large company with business consultants. While focusing on the tools that business can use independently first, it is hard to evaluate SAS as a whole, because they offer so many different packages. Some are tailored for big data and machine learning; others are focused on text analysis. Their base system is SAS Viya, and while it is possible to program in SAS with their own programming language, they also advertise that users can program with R or Python in SAS Viya. Their base programming solution loses out to R and Python, but if one of their pre-programmed solutions does exactly what a company needs then they can indeed be the best choice. Naturally SAS is not open-source and they do not have public pricing, all is based on private quotes. (SAS, 2019)

Apache Spark is an open source computation engine for Apache Hadoop, or other such systems, and to understand the importance of Spark, one must understand the importance of Hadoop. Apache Hadoop is a software library, designed to enable large scale processing of data across clusters of thousands of individual computers, each offering their storage capacity and processing power. Apache Spark allows for the analysis of Hadoop data, with a simple and effective programming language; a user can write applications in Java, Scala, Python, R

or SQL. Spark is well suited for real-time analysis and machine learning. Companies with many computers that want to do computationally heavy analysis without renting or buying powerful servers would be better off with Apache Spark than for example my application, which only runs on a local computer or server and trying to adapt my application to run on clusters is beyond the scope of this thesis. (Apache Spark, 2019)

RapidMiner is perhaps the tool that most directly competes with my application. With a focus on performance and speed, they even allow for integration with Hadoop clusters. They are more like SAS in that it is a for-profit company, but they do not offer as much in the line of consultants. The base RapidMiner system is visual and based on a graphical user interface rather than programming, but they claim it can perform all sorts of analysis. Their Turbo Prep and Auto Model extensions does almost exactly what my application does, only not focused on time series predictions, however getting the version with the extensions can be relatively pricey. RapidMiner also offers a free trial, which I have used to compare with my interface. I will come back to these comparisons the dedicated chapter. (RapidMiner, 2019)

KNIME is another visual programming tool, only it is open source. KNIME has many freely available templates that are designed for many typical analytical applications, such as churn prediction and credit scoring. KNIME is also made to integrate with many other sources, such as Apache Spark. Staring out with one of these templates and modifying them can enable easy custom models and visual programming is much easier to learn than traditional programming. KNIME could certainly be a good option for frugal businesses that can work with public licences and are willing to invest some time in learning a new software, but not in learning a programming language. (KNIME, 2019)

Sisense is made with non-technical users in mind and is much like Tableau in that it primarily focused on visualization and not made for all types of data analysis. Their primary interaction is dashboards and they are also closer to a service in that they offer lots of industry specific premade dashboards. However, it is possible to perform more advanced analytics like predictions by running R formulas. All their pricing is quote based, like SAS. (Sisense, 2019)

I have just shown the tip of the iceberg here, even if it is a highly rated tip. Looking into all these available tools it becomes clear that there is no single right choice for all individual businesses. The requirements on user competence vary wildly and some products are designed with very particular applications in mind, like my interface, even among the free products. It

seems like there could be room for niches of highly specialized solutions. However, what are the choices for businesses that are looking for external professionals to handle the customization and provide start to finish service as well as continual support, and could any such services be affordable enough to target the same companies as my application?

3.2 Analytical services

First, I will go through the market leaders in the field of Worldwide Services Operations Analytic Applications as defined in the International Data Corporation (IDC) report "Worldwide Big Data and Analytics Software 2017 Market Shares: Healthy Growth Across the Board". (Vesset, 2018)

SAS sits at the top with a market share of 17.7% in 2017. SAS Consulting, which is not necessarily what IDC refers to, are focused primarily on the implementation of SAS platforms. They meet with clients to determine their needs and help find and implement what they consider to the best SAS tools. SAS consultants focus on helping clients learn how to use their SAS software in the best way possible. Once the consultants have helped choose and set up the software then the regular SAS customer support takes over. Naturally, SAS is relevant to those who want to use SAS tools. (SAS, 2019)

Next is FICO with a market share of 17.5% in 2017. FICO is more focused on the financial sector, and especially credit as the FICO scores are the dominant credit scores in the US. FICO mostly sell and maintain their own software platforms, but they also collaborate with Tableau so that those technologies can integrate seamlessly. FICO has industry and solution specialized consultant services, such as FICO Fraud Consulting Services and FICO Analytic Consulting. (FICO, 2019)

IBM has a market share of 7.7% but is much larger in other areas. IBM can help clients make entirely custom systems and has the resources to completely redo a client's entire IT infrastructure. This puts them out of reach of smaller businesses and on a different scope than the other tools and services I have reviewed. However, they also offer some of their own analytics software platforms. (IBM, 2019)

There are many more companies, but I believe a useful distinction is whether a company can work any project or if they implement their products. Companies that implement their software

platforms are, for example, DCX, Hitachi, Microsoft, Oracle, and Teradata. Then there are large consulting firms that also deal with any technology implementation such as Accenture, Deloitte, and PwC. As is typical of services that mostly depend on human capital, there are a plethora of smaller consultancies as well. There are websites offering ratings and reviews to make sense of the vast supply of such consultancies, such as Clutch.co and Wadline and then there are matchmaking services to find suitable partners, such as VenturePact and Digitalogy. To illustrate the breadth of this market, I will provide some of those highly rated consultancies.

QBurst is an Indian company with over 1,300 employees across the globe, founded in 2004. They offer a diverse portfolio of programming languages and services, including IoT, Blockchain, and E-Learning. Their clients are mostly small, and mediums sized businesses, but include the United Nations, Rosetta Stone, National Geographic, and Dell. They are relatively cost-effective providers of quality services. (QBurst, 2019)

PSL Corp. is a Colombian company with about 670 employees, founded in 1986. They mostly do custom service development with Java or C# programming, but also cloud solutions in AWS and Azure. They serve medium and large enterprises, including the Fortune 500 companies Brinks, BMC Software, Deloitte, Panama Canal, Arris, and Bridgestone. (PSL Corp., 2019)

SoftwareMill is a Polish company with less than 250 employees, founded in 2009. Their focus is also on custom service development, and they specialize in the Akka, Java, and Scala languages and programming in/with Kafka, Apache Spark and TensorFlow. They work with companies of all sizes, like Tipser, Knip, Zerigo, IP Integrated, Vocado, Attikis, and Intelli. (SoftwareMill, 2019)

Technical talent is spread all over the world, and in many cases, there is no need for programmers ever to set foot in the same country as their clients. Naturally, my contribution is merely an insignificant drop in this sea of talent, but hopefully, a useful example of far one can get with free software in a relatively low fraction of full-time-equivalents. Some of these service providers could clearly be good alternatives to my application. My application is meant to enable data preparation as well as providing a user-friendly way to interact with the AutoML algorithm, and I will present the theoretical background for my work in the next chapter.

4. Theoretical background for the interface

At its core, the act of prediction is merely using historical data to make an educated guess about some aspect in the future. Immediately that highlights a critical factor about prediction; it is wholly dependent on the quality of the utilized historical data, which is reflected in that the average data scientist spends more than 50% of their time on data preparation (Henke, 2019). The second part of predictions is much more complicated because there a multitude of techniques, some more educated than others, for guessing the future.

4.1 Data preparation

Data points in a dataset can be four things, with some overlap, normal, missing, erroneous, or outliers. Depending on the data and prediction technique, one might have to deal with one or more of them. Let me also specify that I define a data point or a value as the value of a single variable and an observation as the values of all the variables for each level of the utilized index. For example, if a dataset contains data on different countries over multiple years, then each observation is a unique combination of country and year while the data points are the values of each variable recorded for a country in a year. Normal data, in this case, means factual data that is not extreme compared to the rest of the data in the dataset. Accurate/correct can be a simplified label for some types of data, especially those that were simplified in the recording, and for those types, normal data means data that resemble their intended reality to the closest degree possible. Missing data is one of the most common problems for data scientists, often shown as not available/applicable or N\A values in a dataset.

4.1.1 Dealing with erroneous data

Erroneous data are data points in a dataset that for some reason do not have the correct value; it could be a measurement error, either human or mechanical, a glitch, corruption in the data file or any number of problems. There is often no way to detect such mistakes by merely studying the dataset, which means one must check a data point by re-recording it, perhaps using a different technique, and compare. If one suspects that there are mistakes in a dataset, it might be prudent to randomly draw a selection of observations and test all of them. Hopefully, the percentage of errors in the sample are the same as in the population. That, naturally, depends on whether the errors are truly random and non-random errors in a dataset are awful for any analysis. Therefore, it is essential to know how the data was gathered and

evaluate how error-prone that gathering was. Some errors in a dataset can be unavoidable, and there will often be a cost/benefit trade-off related to the accuracy of data. Many data scientists will have no way to check the data at all. Some outliers could also be mistakes, and then it can be possible to identify them just by studying the dataset.

4.1.2 Dealing with outliers

If one man has a recorded height of 1185 cm in a dataset with biometrics, that is a clear outlier and a mistake. Perhaps the height was recorded manually, and someone pressed 1 twice when entering the correct height of 185 cm. Another example could be looking at worldwide GDP growth, where Equatorial Guinea grew by more than 100% in 1997. However, that extraordinary growth is entirely true and due to the utilization of significant oil reserves in a diminutive economy (World Bank, 2019). Therefore, there are two types of outliers, those that are clearly mistakes and those that are not, with grey area in between. Depending on the size of the dataset it will often not be feasible to inspect each outlier manually, and if a data point is clearly a mistake, it is not necessarily easy to determine what the actual value is. Thus, if it is possible to formulate a clear rule for what constitutes an obvious mistake, the best course of action is usually to remove those data points, and I will discuss what to do with missing data points in the next section.

If an outlier is not clearly wrong, then studies show that one should not necessarily do anything with it. There are some rules of thumb for when one can remove a true outlier, but the general rule is to leave them be. If an outlier does not affect the results but does affect the assumptions of a model, then it can be removed, but that fact should be communicated clearly, such as with a footnote. If the outliers affect both results and assumptions, then they should not be dropped, but the analysis can be run both with and without them and show how it changes the results. If an outlier creates a significant association, so that it is the basis for the significance of an analysis, then academic consensus is that it should be dropped. Often it is better to transform all the data points of a variable that is troubled with outliers. Log and square root transformations can reign in outliers, but one should not transform data unless it makes sense for all the data points. Handling outliers correctly requires good knowledge of the data and mismanaging outliers could introduce bias in any analysis and makes it vulnerable to justified criticism during peer-review. (Grace-Martin, 2019)

4.1.3 Dealing with missing or deleted values

First, I need to classify missing values more clearly. In general, there are three types, Missing at Random (MAR), Missing Completely at Random (MCAR) and Missing not at Random (MNAR). MAR is when the fact that some data is missing is related to some of the other observed data, but not at all related to the characteristics of the actual missing data. MCAR is when the fact that the data is missing has nothing to with any of the data or the features of the missing data. MNAR applies when the fact that data is missing is related to the value of that potential data, or when it depends on the value of another variable. (Little, 2002)

It is possible to remove observations with MCAR data points without adding bias to an analysis, even though it could severely reduce the size of the dataset. Removing observations with MNAR, and sometimes MAR, data points will tend to add bias. However, imputation, the generation of a new value for the data point, is not necessarily a good option either. There are three ways to remove missing values from the dataset. Deleting the entire observation if there is at least one missing value can be relevant if there are not that many rows containing missing values, but it will add bias unless the missing data points are MCAR and the assumptions for MCAR are rarely satisfied. Deleting the entire variable might add omitted variable bias and is rarely justified unless there is a significant amount of missing values for that variable. Unlike for single observations and datapoints, a proper imputation is almost always preferred over removing an entire variable. The third option is pairwise deletion, which is much like deleting whole observations, but only observations with missing values of interest in a specific part of the model are deleted. So, if a model uses different sets of variables at various stages, then the number of observations at each stage will vary. This deletion will also add bias unless the values are MCAR and can make the model more difficult to interpret. (Little, 2002)

There are many techniques for imputing missing values. However, one should be very careful in choosing among them, for they can severely reduce the quality of an analysis. When imputing values, there are specific time series techniques and general techniques. The simplest methods, such as using the mean, median, or mode of a variable both add bias and deflate the standard errors because they do not include the uncertainty around the correct value. Assigning a new value that is unique and symbolic of a missing value is another simple technique, but doing so requires great care because the chosen value will affect the results in addition to the previous problems.

For time series, it is possible to carry the previous value forward or the next value backward, but again, this adds bias and performs poorly if there are trend or seasonal components. Linear interpolation with seasonal adjustment, if applicable, performs much better, but still artificially reduces the variance. More advanced general techniques are, for example, regression, multiple imputation, and K nearest neighbors (KNN). Regression treats the missing value as the dependent variable and uses the other variables to make a prediction, providing theoretically unbiased estimates. However, this sub-analysis comes with assumptions that depend on the regression technique, which can be a lot of extra work to evaluate, and the predictions will tend to fit together with the other variables, naturally, reducing the variance. (Swalin, 2018)

Multiple imputation uses the observed data points to generate a distribution of possible values and then draws values to replace the missing ones. It is better to use a simulation that includes the uncertainty in the model parameters, such as Markov Chain Monte Carlo simulation. The simulation is done *m* times, creating *m* different datasets and the full intended analysis is run on each of the *m* datasets. Then these results are combined back into a single complete analysis. Many analytical programs include packages for automating most of this procedure. If the distributions used to draw the variables are correct then this provides unbiased estimates and correct standard errors, but automated distributions are rarely accurate, which means this technique can also require a lot of manual evaluation from the analyst. (Rubin, 1987)

KNN uses a specified distance measure to identify the *K* nearest neighbors with observed values and then calculates the missing value as the average of those neighbors. The distance measure and value of K should be based on the type and dimension of data. KNN is simple to understand and easy to implement, and because it is non-parametric, it has an advantage when there is a lot of variance in the data. However, it can be quite time consuming if there are a lot of data, as it searches for the best matches in the entire dataset and the accuracy can fall for highly dimensional data if the difference between nearest and farthest neighbor decreases. KNN and multiple imputation are the most popular ways to deal with missing values. (Swalin, 2018)

I have tried to include these concerns in my application. To help facilitate controlling for errors, I have included an option to extract a custom size subset of randomly selected observations. For outliers, users can explore the data and use simple filters to replace extreme values with blank datapoints. To deal with missing values, I have implemented KNN as the default because it requires less manual modification than multiple imputation.

4.2 Presentation of H2O.ai AutoML algorithm

The AutoML algorithm automates the training and tuning of several different machine learning models, including feature engineering, then ranks all the attempted formulations so that the best model can be used to make predictions. The current version of AutoML trains and cross validates the following algorithms, in the following order: three pre-specified XGBoost GBM (Gradient Boosting Machine) models, a fixed grid of GLMs, a default Random Forest (DRF), five pre-specified H2O GBMs, a near-default Deep Neural Net, an Extremely Randomized Forest (XRT), a random grid of XGBoost GBMs, a random grid of H2O GBMs, and a random grid of Deep Neural Nets. Then AutoML trains two Stacked Ensemble models, one based on all previously trained models, and another on the best model of each family, using cross-validation stacking to produce a single prediction. (H2O.ai, 2019)

AutoML only requires three inputs to begin: the variable to be predicted, a training dataset and at least one of two possible stopping metrics. The stopping metrics are maximum total run time (in seconds) and maximum total number of attempted models. There are several optional data related settings. Users can specify which variables to use for making predictions; otherwise, all variables are used. If the user does not want to use cross-validation, then a validation set can be specified. A leaderboard set can be specified, which is only used to evaluate the performance of each model, and if the user turns off cross-validation, then a leaderboard set is automatically generated from the training set unless it is manually specified. To override how the stacked ensembles are trained, a user can define a blending set, which changes the technique to Blending/Holdout Stacking. By defining a fold column, that column will be used for the cross-validation fold indexing assignment, rather than the default randomized five-fold scheme. The last data related option in R is that users can define the relative importance of each observation by specifying a weights column. (H2O.ai, 2019)

Then there are several options which do not require any reference to data. Users can set the number of folds for cross-validation, which has a default of five. For classification problems the user can choose to balance the classes by oversampling the minority, then specify the over/under-sampling ratios, which are otherwise automatically calculated, and set the maximum relative size of the training data after balancing, which is five by default. Another way to control the run time is to set a maximum number of seconds dedicated to training each model. Users can also specify which metric that should be used for early stopping of models, rather than the automatic default which uses logloss for classification, deviance for regression,

and anomaly score for Isolation Forest. Other early stopping options let the user set the tolerance, or minimum metric improvement for continuing to train a model, which is otherwise calculated based on the size of the dataset, and the number of rounds without significant improvement before stopping, which defaults to 3. (H2O.ai, 2019)

The leaderboard can be sorted as the user wishes by specifying which metric to use, while the automatic default uses Area Under Curve for binary classification, mean per class error for multinomial classification deviance for regression. For reproducibility, the user can specify the seed. However, deep learning models are not reproducible for performance reasons and should, therefore, be excluded if reproducibility is required. The users can exclude all the algorithms they want, or define the only algorithms to be used. Finally, there are three more cross-validation options. Users can elect to save the cross-validation predictions, which is a requirement for running stacked ensembles more than once but defaults to false. The cross-validation models can also be saved, although that will consume a significant amount of memory in the H2O cluster and defaults to false. Cross-validation fold assignments can be saved as well. (H2O.ai, 2019)

In my application, which must be entirely agnostic about the data supplied by the user, I only specify the required options, choosing maximum total runtime, and then a random 20% leaderboard set. I specify a leaderboard set to utilize the cross-validation functionality for everything except the model scoring, as an extra precaution against overfitting, because the models do not have access to the leaderboard data during their training.

Other data scientists have tested and evaluated the quality of H2O AutoML. Adithya Balaji and Alexander Allen have extensively tested some of the most popular automatic machine learning algorithms in their 2018 paper, “Benchmarking Automatic Machine Learning Frameworks.” In it, they selected Auto_ml, a TensorFlow and Keras based package, Auto-sklearn, a Linux based system, TPOT or Tree-Based Pipeline Optimization Tool, another Python program, and H2O, which is programmed in Java. For classification problems, the ranking was, in descending order, Auto-sklearn, TPOT, H2O, and Auto_ml. For regression problems, the ranking was, in descending order, TPOT, H2O, Auto_ml, and Auto-sklearn. Further, H2O was found to be the most resource-intensive. While, not at the level of proficient human data scientists, all models were capable of providing relatively accurate predictions of good quality. (Balaji & Allen, 2018).

Erin LeDell, the Chief machine learning scientist at H2O.ai., has tested AutoML on Kaggle[6] herself, with all her competition results available on her page on the Kaggle website. In the KaggleDays SF Hackathon, AutoML ranked 8 out of 74 participants, using only one line of code and a runtime of 100 minutes. (LeDell, 2019)

AutoML is advertised both as a tool for novices and experts since it allows a high degree of customization. Because AutoML is written in Java and has an excellent R programming API in addition to its high performance, it was a clear choice for me.

[6] Kaggle is a website for hosting data science competitions where there are often significant cash prices to the best contributors.

5. The Interface[7]

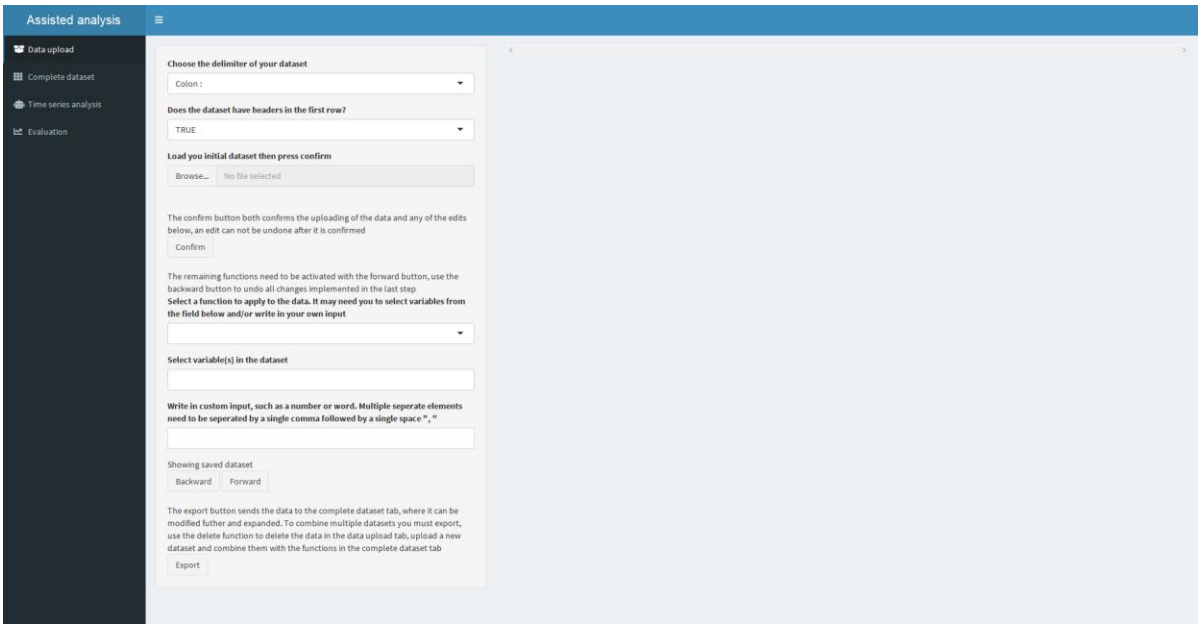


Figure 5-1 The screen that users see upon startup of the interface.

The main part of this thesis has gone into programming an application in the R programming language using the **shiny** package. **Shiny** offers simple functions that help transform R code into HTML code and allows the user to interact with the R code through a browser application. **Shiny** is focused on making aesthetically pleasing and easy to use applications that can run on a local computer or a server. (Chang et al., 2019)

Conceptually, the interface is designed to process multiple machine-readable datasets from different sources into a single dataset and make predictions into the unknown future for one of the variables, and do so in such a way that someone without any experience with predictive analysis can complete it themselves. To accomplish this, I have utilized many different R packages, but the prediction function is entirely based on the Automatic Machine Learning (AutoML) algorithm made by H2O.ai. Thus, my R code is another interface into H2O.ai's Java core. H2O has its own browser-based interface, Flow UI, but through simplification and

[7] In this more technical section of the thesis, the concrete names of R packages will carified with **bold** marking and actual function names will be in `monospace font`. This will be especially noticeable for the word H2O, which will both be used to refer to the company and the R package, as shown with **bold** marking.

specialization, I believe my interface is better at its intended purpose than the more general Flow UI. (H2O.ai, 2019)

My application is far from a universal miracle program, as I have made many restrictions due to time and workload constraints. The interface can only generate meaningful predictions for time series, and while it is possible to upload and process panel data, a single entity must be selected before predictions can be made. I chose to focus on time series and predictions into the actual unobserved future because the setup for such analysis is much more complicated than, for example, classification or regression based on incomplete cases. There are many freely available interfaces, such as Flow UI, that should be sufficiently intuitive for novice users to accomplish such analysis. Predictions into the actual unobserved future also seems to be one of the most useful forms of analysis for decision-makers, as well as one of the most under-utilized forms of data analytics in the private sector (Henke, 2016).

I will use two simple examples to clarify what I mean by predictions into the actual unobserved future and how it differs from the analysis of incomplete cases. A company that wants to predict the performance of one of its subsidiaries 2-5 years from now has no data from those 2-5 years, and therefore the predictions are into the unobserved future. A company that wants to predict whether a customer will default on their loan by analyzing their current data on all customers with recorded outcomes is filling in an incomplete case. For such problems, the time dimension is irrelevant, and the techniques applied do not discern whether the outcome is missing because it is in the future or because it is an error in recording.

When creating models that aim to predict outside the scope of their observations, it is important to avoid training the models on data they would not have if they were making future predictions. Further, it is not necessarily trivial for novice users to get the actual predictions for periods outside the dataset. It is these things my interface handles, in addition to presenting results in an understandable format, while leaving the model creation and parameter tuning up to autoML. However, most of my time has been invested in enabling the uploading, processing, and combining of different datasets.

The majority of my functions are selected from a dropdown menu to make interacting with the data as simple as possible. Variable input is selected from another menu that is connected to the actual data, custom input is typed into a text box and the function is activated with a forward button. The user can then visually inspect whether the function had the desired effect

by looking at an automatically rendered table of a section of the data and confirm the change with a confirm button. The functions are designed to do nothing, rather than crash the application, in the event of incorrect usage, while displaying the error message in the connected R window.

5.1 Presentation of the application structure

There are a few lines of code outside of the actual **shiny** application, which are simply for loading all utilized packages. Individual packages will be explained where they are used. However, the **pacman** package is used in this section to load most of the other packages, as it improves consistency and speed (Rinker et al., 2019). Three packages, **H2O**, **fliptime,** and **imputation** are loaded outside **pacman** because it did not load the most recent version of **H2O** and **pacman** had trouble with **fliptime** and **imputaion**'s GitHub source. As such, the GitHub packages were loaded with the **remotes** package (Hester et al., 2019).

The code is designed to install everything and run the application without any user modification, enabling easy installation, albeit with many downloads for users that do not have most of the packages already. However, there is one potential problem that must be handled manually by the user. **H2O** requires an updated 64-bit JDK or JRE version of Java, which might not be installed, but is freely available at Oracle's website: https://www.oracle.com/technetwork/java/javase/downloads/index.html

Once the packages are in order, the code for the actual **shiny** application begins, and all further code is written inside the **shiny** frame. The **shiny** application consists of two parts, the UI and the server function. The UI naturally deals with all user inputs and the outputs shown to the user. The UI programming is simple, inputs and outputs need to be accurately named, but otherwise one merely chooses where and what to render to the user. That is, in fact, one of the key strengths of **shiny**, it is easy to make nice looking applications. The server function is where developers program how inputs are processed into outputs and is much closer to normal R programming. A key difference is that outputs must be handled by **shiny**'s functions, and it is important to consider **shiny**'s reactive environment.

A key part of **shiny**'s functionality is this reactivity, and it is achieved by creating a cascading network of invalidation connections that make related functions re-calculate when any of their "family" is changed. To prevent everything from recalculating whenever the user does

anything one can isolate parts and store some objects outside of shiny's reactive environment. Initially, I had significant difficulty with finding a good way to implement iterative changes to uploaded data, which is something I will elaborate later.

5.2 User Interface design and server function

The UI is formatted as a dashboard, based on the **shinydashboard** package (Chang et al., 2018). As a dashboard, the UI is split into a header, sidebar, and body. The header is always present, contains the title of the application and a button to toggle the sidebar. The sidebar lets the user navigate to different dashboard bodies, which contain the inputs and outputs.

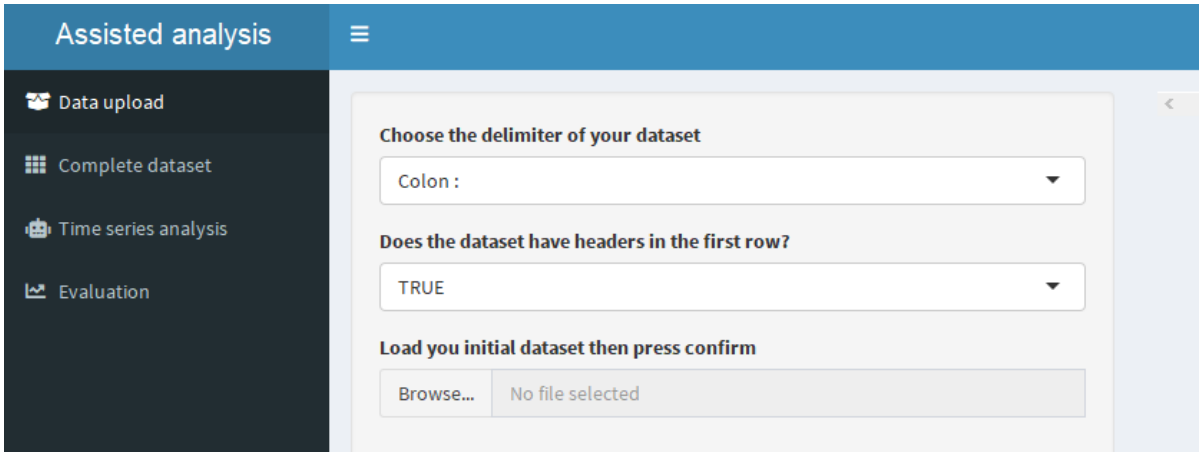


Figure 5-2 Closeup showing the complete functionality added by the dashboard configuration

In Figure 5-2 the dashboard sidebar is shown in black, the header in blue, with the title in the top left and the button to toggle the sidebar just to the right of the title. The body of the first sidebar element or tab is the remaining area in light blue and grey. My application has four different areas that are highly interconnected, and the user is supposed to go through them from top to bottom. I chose to use this dashboard configuration and four-way split rather than a single body to scroll down because the application has four distinctly different processing elements in it. Keeping each element in a separate tab makes it much easier to limit the information and interactions shown to the user to what is immediately relevant.

The data upload tab is for uploading raw datasets and modifying them. The complete dataset tab is for combining different datasets and performing the final preparations for the machine learning, while also including all the modification options available in the data upload tab. Once all conditions in the complete dataset tab are satisfied, the user may continue to the time

series analysis tab. The time series analysis tab has much less user interaction, as it is just designed to initialize the machine learning and display the essential results to the user. The evaluation tab is for more advanced users and is full of performance metrics and information regarding the machine learning models that have been applied to the data.

The key design feature of each tab is that the user should only have to interact with the programming backend through dropdown menus, buttons, and occasionally write in custom input in response to a predefined function. Through this limitation and clear instruction, the interactions should be intuitive and help prevent user error. Naturally, it is also impossible to do something that has not been explicitly coded into the application, which should not be much of a drawback, as the intended users are those without programming experience. By allowing more custom input, such as users writing in which functions to use, the application could be much more flexible, but would also be much closer to programming, reducing its value compared to programming directly in R.

The server function is what makes the application work, and I will explain how everything works and my reasoning behind the noteworthy solutions. Almost all the code is directly related to one of the tabs. However, the **H2O** initialization and max upload capacity are set immediately. I initialize **H2O** immediately rather than right before it is utilized because it can take a few moments to connect and if there is something wrong with the **H2O** requirements or setup then the error message should appear before the user has invested time into application. I set the max upload size to 100 MB because that should be more than enough for any single text-based dataset, but it is straightforward to adjust.

5.2.1 The Data upload tab

All the different dashboards bodies are themselves separated into a sidebar, and main body, where the sidebar is for user input and main body is primarily for rendering outputs.

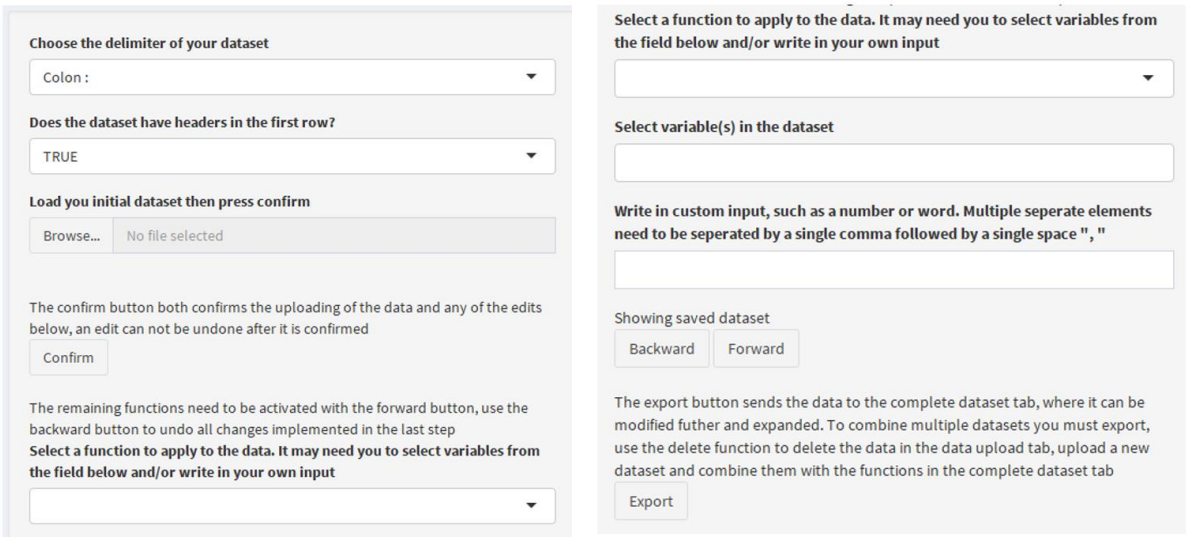


Figure 5-3 The sidebar panel inside the body of the data upload tab

The data upload tab handles the uploading of datasets and initial processing. The first element in the sidebar lets the user select the delimiter of their dataset, as shown in Figure 5-4.

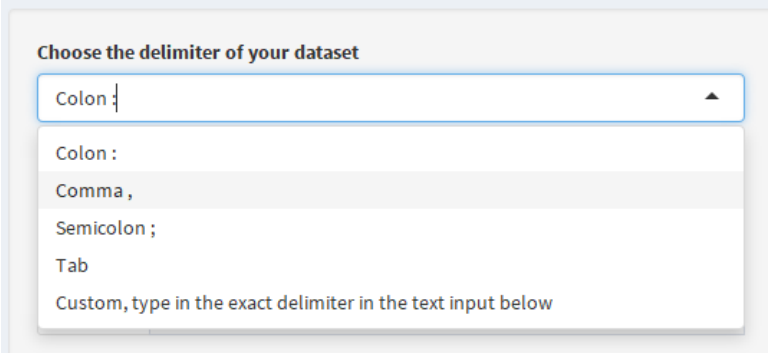


Figure 5-4 The delimiter dropdown menu

The dropdown menu contains four typical delimiters that when selected, provide all the necessary information to the data upload function. If the custom option is selected, then the user must type in the exact delimiter in the general text input further down in the sidebar before confirming the upload. Next is the simplest user input in the entire application, the header dropdown menu in Figure 5-5.

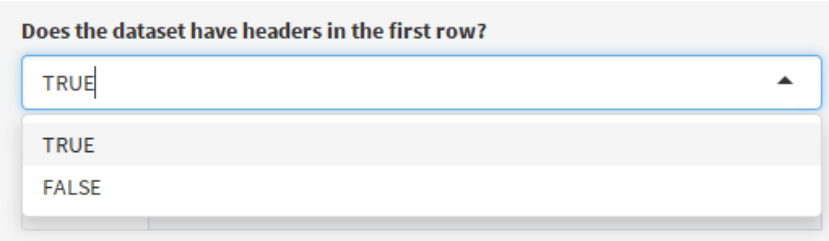


Figure 5-5 The header dropdown menu

In this menu, the options are TRUE and FALSE because those inputs can be used directly in the code and are just as understandable as yes and no. Here the user decides whether the first line in the dataset should be considered as headers, but there are options to remove rows and set headers later as well, so if the data is in a non-standard format it is still not a problem.

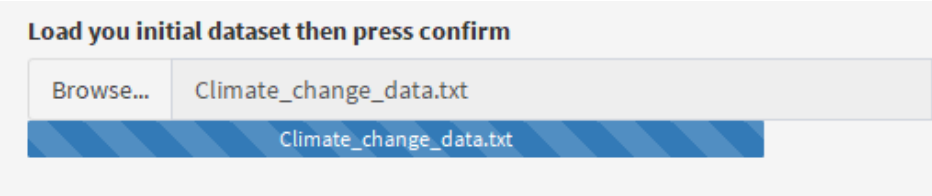


Figure 5-6 Dataset upload button

Figure 5-6 shows the dataset upload button, which initializes a standard browser file upload interaction, where the user can navigate their file structure, select the desired data, and click ok. The data will load with a progress bar, usually close to instantly, and the name of the file is displayed clearly. The data is saved as a table immediately, but nothing is rendered to the user until they press the confirm button, as shown in Figure 5-7. This is part of the system for iterative changes.

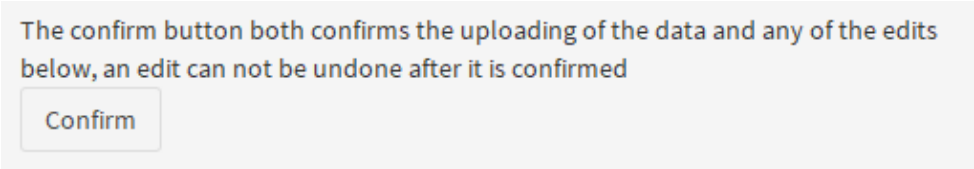


Figure 5-7 The confirm button

Once the user presses the confirm button a table based on their choice of delimiter and header option will render in the main body, as shown in Figure 5-8.

The data upload is a single function. I use the `read_delim` function from the **readr** package because it offers improved speed and flexible detection of data types in each column (Wickham et al., 2018). `read_delim` takes a delimiter and header argument, in which I refer

to the user inputs and some extra values that should be interpreted as missing data. The results of the function are stored in a global dataset object and linked to a reactive object that updates each time the Confirm button is pushed, a reactive shadow so to speak. It is the reactive shadow that is rendered to the user.

Show 10 entries Search:

	Country.Name	Date	GDP.growth	Population.growth	Aid.per.capita	Tyrant	Civil.War	Authoritarian	Flawed.Democracy	Oil.(
1	Algeria	1961-01-01	-13.60544133	2.485473208	38.3967921	0	1	1	0	
2	Algeria	1962-01-01	-19.68504183	2.470737085	34.01409716	1	1	1	0	
3	Algeria	1963-01-01	34.31372878	2.492035183	23.52163547	1	1	1	0	
4	Algeria	1964-01-01	5.839413007	2.560435108	18.52964833	1	1	1	0	
5	Algeria	1965-01-01	6.20689821	2.656201066	11.58078371	1	1	1	0	
6	Algeria	1966-01-01	-4.804970943	2.75967041	9.777919052	0	0	1	0	
7	Algeria	1967-01-01	9.452962563	2.840043629	8.060387307	0	0	1	0	
8	Algeria	1968-01-01	10.79623857	2.879980513	8.702461594	0	0	1	0	
9	Algeria	1969-01-01	8.433280282	2.869095002	9.277851831	0	0	1	0	
10	Algeria	1970-01-01	8.862657109	2.827185777	8.399293088	0	0	1	0	

Showing 1 to 10 of 57 entries Previous 1 2 3 4 5 6 Next

Figure 5-8 Table rendered to the user

There are five essential features to the table, other than simply displaying the data. In the top left the user can choose how many rows to show at the same time, though limited to either 10, 25, 50 or 100, shown in Figure 5-9. In the top right there is a search bar. This search bar filters the data so that only rows with at least one column containing the search term are displayed. The term is matched as if all the information was plain text, meaning a plain number can be matched to any of the decimals in an entry.

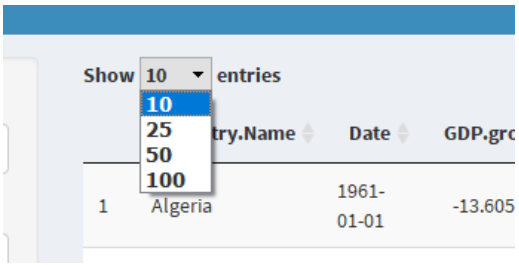


Figure 5-9 Table number of entries selection

Also visible in Figure 5-9 are the triangle indicators next to each variable name. By clicking the name of a variable, the table will be sorted according to the values of that variable, while the tringle indicators show whether it is sorted an ascending or descending order. Directly underneath the table, there is text showing which rows are being displayed as well as the total number of rows and buttons for navigating the sections of the dataset. Finally, there is a bar at the very bottom that can be dragged to move the display to show more columns while hiding others, just like a standard display bar.

Rendering a table to the user can slow down processing times if it is a big dataset. However, this is mitigated by only showing a small section of the data at any one time. A table is necessary so that the user can evaluate whether the changes they are implementing succeed or if there is an error and the sorting and search functions are very useful for exploring the data and looking for outliers.

The table in the main body of the tab is rendered using the `DT` function from the **DT** package, which is designed to work with HTML tables and is what enables all the extra user interactions. The scrolling bar underneath the bottom of the table is simply an added style option for dealing with overflow. (Xie et al., 2019)

To begin processing the data, the user can choose from an alphabetically sorted list of 17 functions, some of which are variations of the same function. These functions are available in the dropdown menu shown in Figure 5-10. Some functions require the user to select one or more variables, and some require very simple custom inputs. All functions are activated with the Forward button, and none of their changes are saved until the user presses the same Confirm button used in uploading the data. A user can confirm a change and implement a new function without ever clicking the Backward button, but when the Backward button is clicked the rendered table shifts back to the last saved version of the data.

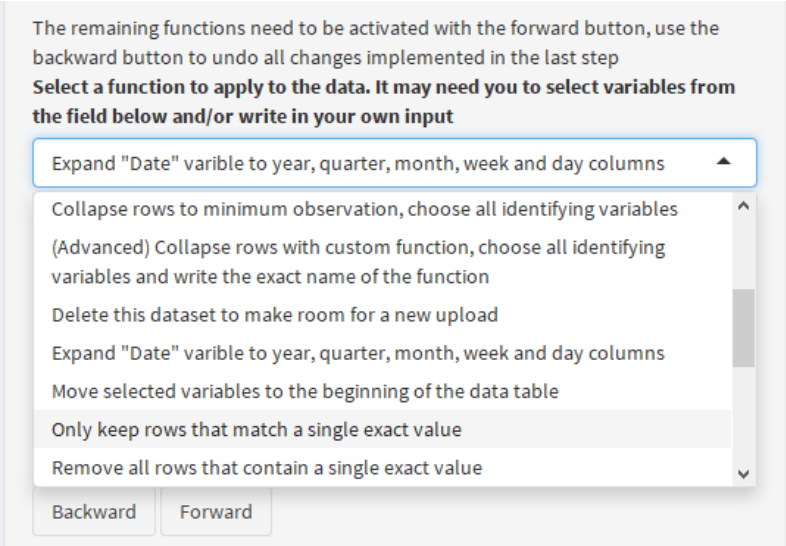


Figure 5-10 Function selection menu

All changes to the data are implemented in a separate reactive object, a change dataset, that activates each time the Forward button is pressed. Through if statements, the change dataset is only ever one modification of the current global dataset and the Forward button simultaneously switches which dataset is rendered to the user.

If there is data in the change dataset, and the change dataset is on display, clicking the Confirm button uses that data to overwrite the global dataset object, in addition to updating the reactive shadow with the new values. This solution enables the user to perform iterative changes with the ability to evaluate the effect at each step. By using a global object as an intermediary between the two reactive objects, I avoid infinite recursion. Initially, I tried only to use **shiny**'s reactive objects and moderate the connection by isolating them from each other. However, isolating a reactive object only prevents it from recalculating when its sub-objects change and using requirements to prevent recalculation prevented any changes from being saved at all. In the end, I have used many global objects, as that works well with **H2O**, which is not designed with shiny in mind.

As previously mentioned, the Forward button is used to activate all functions and switch between which table is rendered to the user, while the Backward button activates no functions and only switches between tables. The way this works is that a run number defines which dataset is shown to the user. Each time the Forward button is pushed the value of the run number is set to one while pushing the Backward button sets it to zero. Since there is no way to show the change dataset without clicking the Forward button and thus generate a new

change dataset, and the Confirm button only saves the changed data if that is being displayed, the Backward button is effectively an undo button.

To select variables in conjunction with one of the functions the user must choose from the variable selection field shown in Figure 5-11.

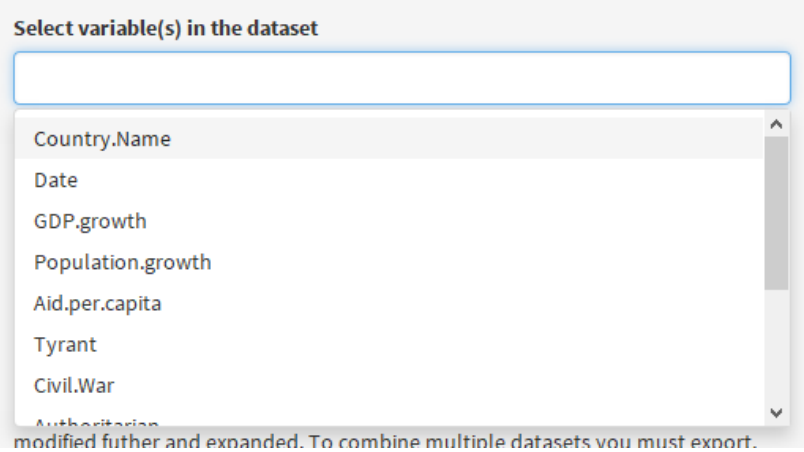


Figure 5-11 Variable selection field

The variable selection field is a link to the actual dataset, and the user can select one or more of the actual variables by clicking on them. This link is achieved with **shiny**'s `updateSelectInput` function, which updates whenever the dataset changes and therefore, always gives the current variable names. Chosen variables are stored as a character vector that is utilized directly in dependent functions. The custom input field is simply a blank field in which the user can type in a number or word that is stored as a single character. Therefore, dependent functions that need numbers transform the character first and those with support for multiple inputs split the character into a character vector by using strsplit and the ", " delimiter, a single comma followed by a single space. Not all functions accept multiple inputs, nor do all functions work on multiple variables at the same time. The descriptions should make it clear what inputs each function needs and accepts, but in the event of a misunderstanding, the user can try again if nothing happened or click the backward button and try again if a function had unintended effects on the data.

Now I will quickly go through the different functions in the Data upload tab and why I included those.

"Add a new variable to the dataset, you can set a value for all observations by writing it"

This function saves the users single custom input as new.var and column binds it to the end of the dataset. The column is then named new.var, but the user can use another function to rename

it and move it to the start of the dataset if so desired. I added this function primarily to accommodate the combination of multiple datasets. If historical data is stored in different files, then new variables might be added or removed over time, because the datasets must have the same variables to be combined with row binding this function lets the user add a variable and potentially set its value as missing. It can also be used to create an identifier for which dataset each observation comes from.

"Collapse rows to X, choose all identifying variables."

There are six different functions with this format to enable different functions for collapsing rows. I have included mean, maximum, minimum, first and last observation as standard functions and added the option to write in a different function manually, which is marked as an advanced option because that requires the user to know the exact name of an R function. These functions were added to enable the combination of datasets with different time horizons. If a user's primary data is quarterly and they want to add data from a daily dataset, then they can collapse the date with each rows quarter as one of the identifying variables. All of the functions are based on the **dplyr** package's `group_by_at` and `summarise_each` functions (Wickham et al., 2019).

"Delete this dataset to make room for a new upload"

This function sets the change dataset to null and if confirmed that sets the global dataset object to null as well. I included it to give the users a cleaner way to upload new datasets. However, it is not strictly necessary because it is sufficient to press the Backward button, upload a new dataset, and then confirm it.

"Expand "Date" variable to year, semester, quarter, month, week and day columns."

Designed to go after a different function in which the user selects the variable that will be transformed into the "Date" variable, this function creates separate columns for all the standard periods at the same time. The user can easily delete the surplus variables, and I chose this way because it avoids asking for a custom input and making the function selection menu more crowded. Given that the dataset has a correctly formatted Date variable, the function uses **dplyr**'s `mutate` function along with `lubridate::year`, `lubridate::semester`, etc. The **lubridate** functions allow for efficient automatic detection of time period, given a

date variable (Spinu et al., 2018). With the period identifiers, the user has sufficient information to collapse the observations to their chosen period.

"Move selected variables to the beginning of the data table"

Merely changes the order of the columns in the dataset and is entirely cosmetic. It changes the order of the columns to put the user's selection first and then the rest after. It might be a little awkward if the user wants to redo the order of columns to a specific pattern, but cosmetic changes are not the focus of this application.

"Only keep rows that match a single exact value for selected variables"

Based on **dplyr**'s `filter_at` function, it removes all rows that do not include the user's custom input in the selected variables (Wickham et al., 2019). The function lets the user remove all unnecessary observations quickly, such as for choosing a specific entity from panel data.

Next, are three very similar functions, the "Remove rows with values greater than limit for selected variables," "Remove rows with values less than limit for selected variables" and "Remove all rows that contain a single exact value for selected variables."

These are also based on **dplyr**'s `filter_at` function, but the user defines what to remove rather than what to keep (Wickham et al., 2019).

"Rename selected variable(s)"

This function allows multiple custom inputs, as long as they are separated by a single comma and a single space. Using `grepl` to determine which variables were selected, the names of chosen columns are overwritten with the custom inputs. The function only detects which variables were selected and not the order they were selected. Therefore, the user must write in the new names in the same order as the columns appear in the dataset from left to right. The ability to rename columns is crucial, as combining datasets is heavily reliant on identical variable names.

"Remove selected variable(s) entierly"

Anther simple function. It uses `grepl` and only keeps variables that do not match the selected variables.

"Select the date variable, which will be renamed “Date”"

This is an essential function. The user selects the single variable that best represents the dates in their dataset, and then **flipTime**’s `AsDate` function automatically detects what kind of format it is in and changes it to the standard R date format. **flipTime** was the most flexible date recognition package I could find, which is necessary because the users can upload datasets with a host of different date formats. **flipTime** is stored on GitHub rather than CRAN (Displayr, 2019). I chose to automatically rename the variable to “Date” because that simplifies many of the later functions, the downside is naturally that the user could cause problems if they decide to rename it to something else anyway.

"Transform data from wide to long form, select possible ID variables from list"

The last function on the data upload tab is one of the more complex, but luckily, it is easily handled by the `melt` function from the **reshape2** package (Wickham, 2017). `melt` only requires a dataset and a character vector with variables, which is easily chosen by the user. The data must be in long format, meaning different observations by row and variable names as columns, for the machine learning analysis, which is why I only include this option and not changing from long to wide form.

The last part of the data upload tab is the Export button, as shown in Figure 5-12.

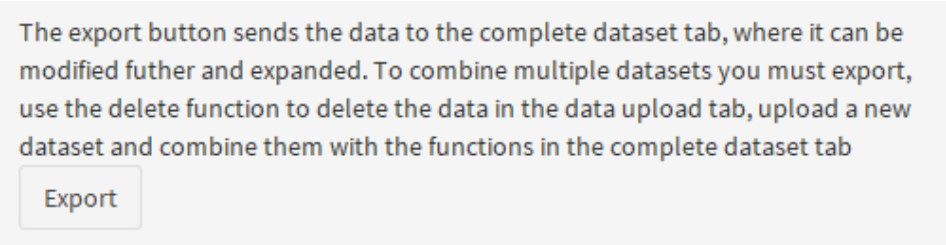


Figure 5-12 Export button

Once the user is finished with the initial data processing, the data can be exported to the complete dataset tab for more processing and final preparations for time series predictions. By exporting the data, deleting it and uploading new data, multiple datasets can be combined in the complete dataset tab. The Export button only saves the current global dataset object to a new global dataset object that the Complete dataset tab interacts with in almost the same way as the Data upload tab interacts with the data upload button.

5.2.2 The Complete dataset tab

The complete dataset tab is very similar to the data upload tab, as is clear from the sidebar shown in Figure 5-13.

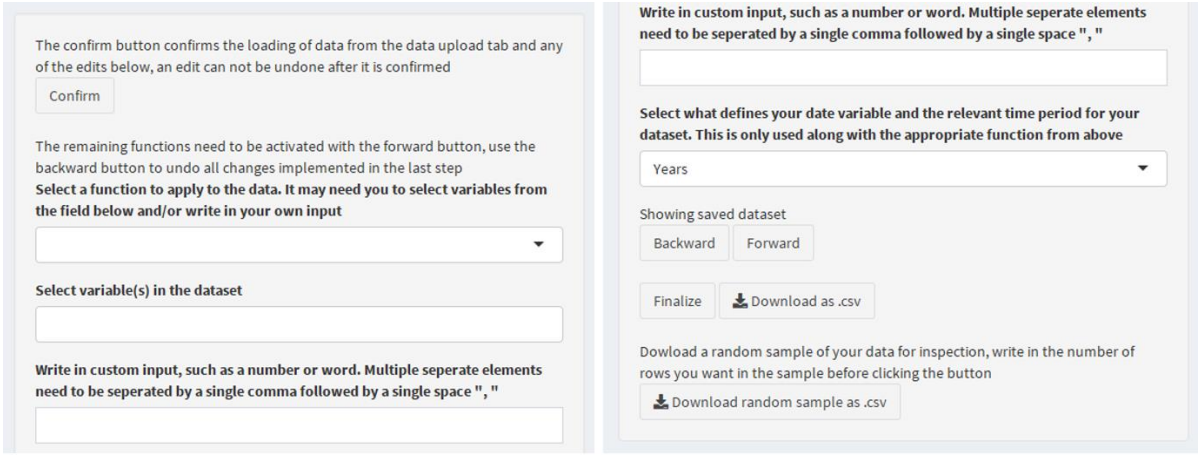


Figure 5-13 Complete dataset tab sidebar

The data upload inputs are gone; there is a new period definition selection menu, a Finalize button rather than an Export button, and two download buttons. The function selection contains 14 new functions, also alphabetically sorted, as well as all the functions from the data upload tab, but the two groups are kept separate with distinct headers in the dropdown menu. The period definition selection function, shown in Figure 5-14, is only used with one specific function and includes all standard period definitions. The model is most likely too slow to work well with data on a higher frequency than days.

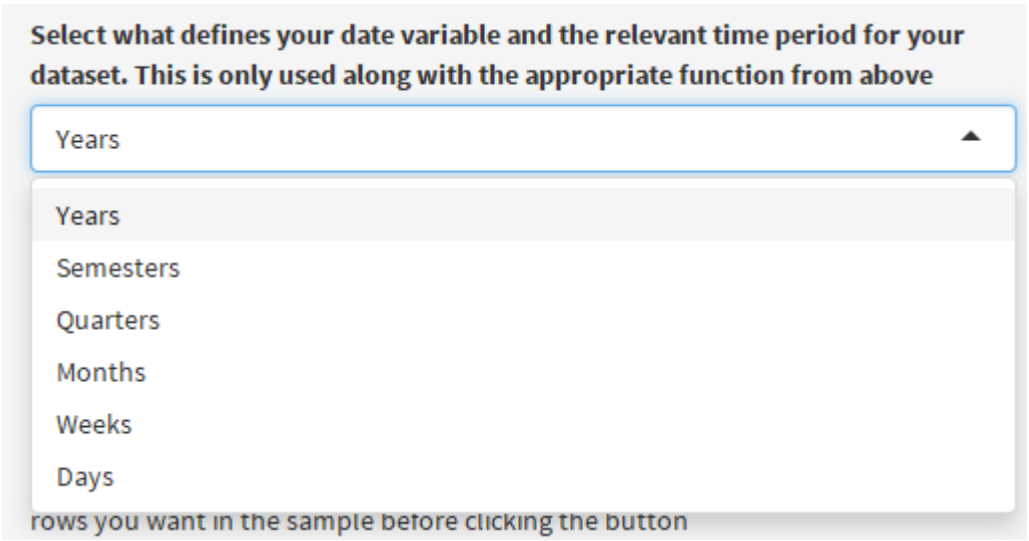


Figure 5-14 Time period definition selection

The data table presentation in the main body of the complete dataset tab is the same as before. However, underneath the table, there are a series of conditions, along with guidance, for what functions must be used before the data is ready for time series predictions, shown in Figure 5-15.

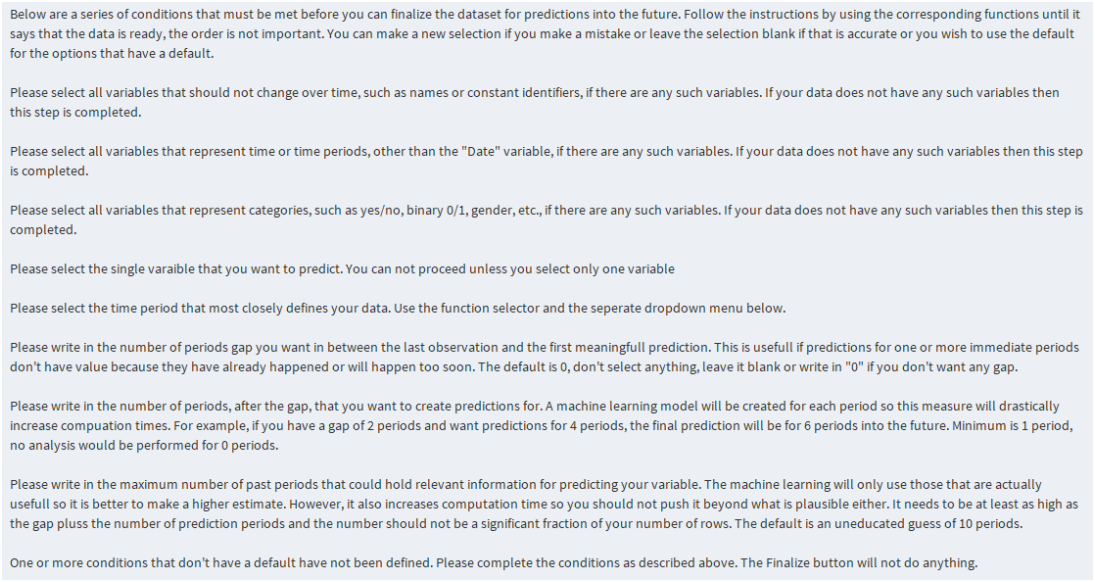


Figure 5-15 Time series prediction guidance and conditions

The text updates with the user's selections and displays an error message if there is a mistake. Many of the functions set values for the machine learning algorithm, and most do not directly influence the loaded data table. Therefore those functions do not have to be confirmed, just initialized with the Forward button. Once all conditions are satisfied, it will say so at the bottom of the main body, and the user may click on the finalize button to send the data to the time series analysis tab. The condition objects are all created and set to null values when the application starts to avoid errors with non-existing objects. That function only sets values for the objects that do not already exist, and after activating once it turns off for the rest of each user's session.

The new functions in the Complete dataset tab are:

"Add the rows of a new dataset to this dataset, both datasets must have the same number of variables and those variables must have identical names. Use other edits first if that is not the case"

This function is the simplest way to combine datasets. If both datasets have the same variables, then the new dataset is row bound to underneath the complete dataset.

"Add the variables in a new dataset to this dataset, please select the identifying variables, such as name and date, and those variables must be identically named in both datasets"

Probably the most useful function for combining different datasets, this function needs a set of variables that are enough to uniquely identify each observation in the datasets and then imports all variables other than the identifying variables from the new dataset in the upload tab to the complete dataset. This is done using the `merge` function, set to keep all observations of the complete dataset by loading missing values for the rows that do not have a match in the new dataset. I chose to keep all observations of the complete dataset, and not generate new rows for the observations without a match from the new dataset because I assume that the user uploads the prime dataset first and then adds additional information. The function can handle one to many merging as well so that if for example, the complete dataset is quarterly information and the new dataset is yearly information, each quarter of the same year will get the same values after the merging.

"Fill in all missing numeric values in the dataset using KNN imputation. This function might not work if you don't have enough complete data"

This function is the most computationally heavy of all the functions, not including the machine learning, and is an analysis on its own. Luckily it is easily implemented with the **imputation** package's `kNNImpute`, which chooses appropriate distance metrics automatically and fills inn all missing values in the dataset (Wong, 2013). The value of K is defaulted to 10, which is a reasonable rule of thumb and choice for most datasets with some size (Swalin, 2018).

The "Replace all of an exact value with NA's for the chosen variables", "Replace all values greater than or equal to limit with NA's for the chosen variables" and "Replace all values less than or equal to limit with NA's for the chosen variables" functions work almost exactly like the remove functions from the data upload tab. Except that they use `mutate_at` rather than `filter_at`, to replace individual values with NA's. These functions are there to let the user deal with outliers if they deem that necessary, such as if an outlier is obviously a mistake. Since the machine learning models cannot use any rows with missing values, the user needs to use the KNN imputation function after they are done removing values.

All of the remaining eight functions are not edits to the data, but the recording of values that might be necessary for the correct formulation of the time series analysis problem. Each of those functions corresponds to one of the conditions in the main body and require either a

variable selection or a custom input. Therefore, I will discuss the function and the related condition together. All the conditions are based on global variables with reactive shadows; this was necessary to prevent the values from erasing themselves and make sure they were available to the H2O functions that need them.

"Select all variables that should be constant over time in your data, such as names"

The constant variables need to be selected because they should not be lagged and should be included in the prediction presentation for the added information. A null value here is acceptable as it is perfectly possible that the user does not have any constant variables in the dataset. The related condition shows this general information when there is a null value and shows the user's selection(s) if there is any.

"Select all time related variables except the "Date" variable"

All time variables except the Date variable are removed before the machine learning because the models use a special time series signature augmentation, and there is no need for overlap with any of the other time variables. Otherwise works like the previous selection function.

"Select all categorical variables"

Categorical variables need to be formatted as factors for the machine learning to treat them correctly, especially if the variable to be predicted is categorical. Otherwise, it works like the previous selection function.

"Select the variable you want to predict"

The models can only generate predictions for a single variable, and there must be a variable to predict for there to be any analysis. Therefore, the related condition instructs the user to pick only one variable when there is a null value, reminds the user that they can only predict one variable and need to make a new selection if the user selects more than one, and shows the user's selection when they pick a single variable. The Export button does not function unless this condition is satisfied.

"Select the time period that best describes your data. Choose from separate menu below"

This is the only function that uses the new period definition menu. The period definition is used to generate the dates for the future prediction periods and are relevant for the quality of

the model to the extent that time periods hold important information. The Export button does not function unless this condition is satisfied.

"Write in the number of periods gap you want between the last observation and the first useful prediction"

Here the function requires that the user types in an integer in the custom input field. The related condition has four outcomes, no value, in which the default of 0 is used, a non-integer input, which displays an error message, a negative integer input, which also displays an error message and a positive integer input, which is displayed to the user.

"Write in the number of periods into the future you want to predict"

This function works like the gap function, except the default is 1 and an error is shown for all integer inputs of less than 1.

"Write in the maximum number of past periods that could be relevant for predicting your variable"

This function also works like the gap function, but the default is 10 and if the value is too low then the number of relevant past periods is set to the sum of the gap and prediction periods.

Since the condition functions do not edit the data, they do not have to be confirmed, but if the user makes a mistake, inputting a blank value or no variable selection resets the condition. Otherwise, the user can also input a new value or variable to overwrite the previous selection.

The Download as .csv button starts a standard file download procedure, which is subject to the local browser’s standard settings and lets the user download the entire, complete dataset and set the name plus .csv. The Download random sample as .csv button works in the same way, except that the user needs to define how many rows they want in the custom input section, with a number, before clicking the button. The random subsample is meant to facilitate checking the data for errors manually while avoiding the bias of just looking at the entire dataset oneself.

The Finalize button does nothing until the user has selected a variable to predict and a period definition because those do not have any natural defaults. If activated, the button is very similar to the Export button. However, the finalize button also initializes functions that set the actual

gap, prediction periods, and number of lags to be used in the machine learning. The actual values are based on the user's inputs or lack thereof. The gap is set to 0 unless the user has specified a positive integer and the prediction periods are set to 1 unless the user has specified a strictly positive integer. The number of lags is the maximum of the user's positive integer input and the gap plus the prediction periods, if no correct input is given, then a default of 10 is used rather than the user input. Further, a sequence of numbers from 0 plus the gap to the gap plus the number of prediction periods is created, because that sequence is the number of prediction models to be made and which lags each of the models can use. Then a sequence of date values based on the user's period definition and lubridate's time period functions are created, which correspond to the final observed date value extended according to the definition for a number of periods equal to the gap plus the number of prediction periods.

A sequence of numbers from the number of rows at the end of the observed dataset to the end of an extended dataset is also created, to make it easy to refer to the actual predictions in the extended dataset. Then a new global dataset object is created and extended with the new date values. Constant variables have their value set to the value they have in the first observation, and the rest are left as missing values.

5.2.3 The Time series analysis tab

The tab where the actual machine learning is initialized does not have nearly as many opportunities for user inputs, as everything should have been prepared in previous tabs. However it still has a main body and a sidebar, the last of which is shown in Figure 5-16.

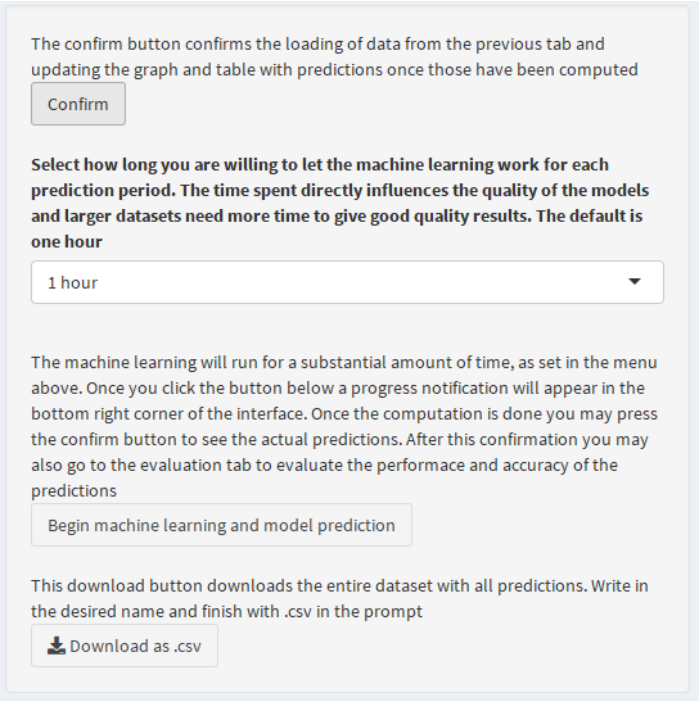


Figure 5-16 The Time series analysis tab sidebar

With only three buttons and a simple drop-down menu, this tab is mostly for starting and showing the key results of the machine learning. The confirm button works as before, updating the reactive objects shown to the user with new information. Clicking “Begin machine learning and model prediction” initializes a whole host of functions that result in actual predictions for a user-defined amount of time periods after the last observation in the data. After data is loaded from the complete dataset tab, but before the machine learning is finished, the main body looks like what is shown in Figure 5-17.

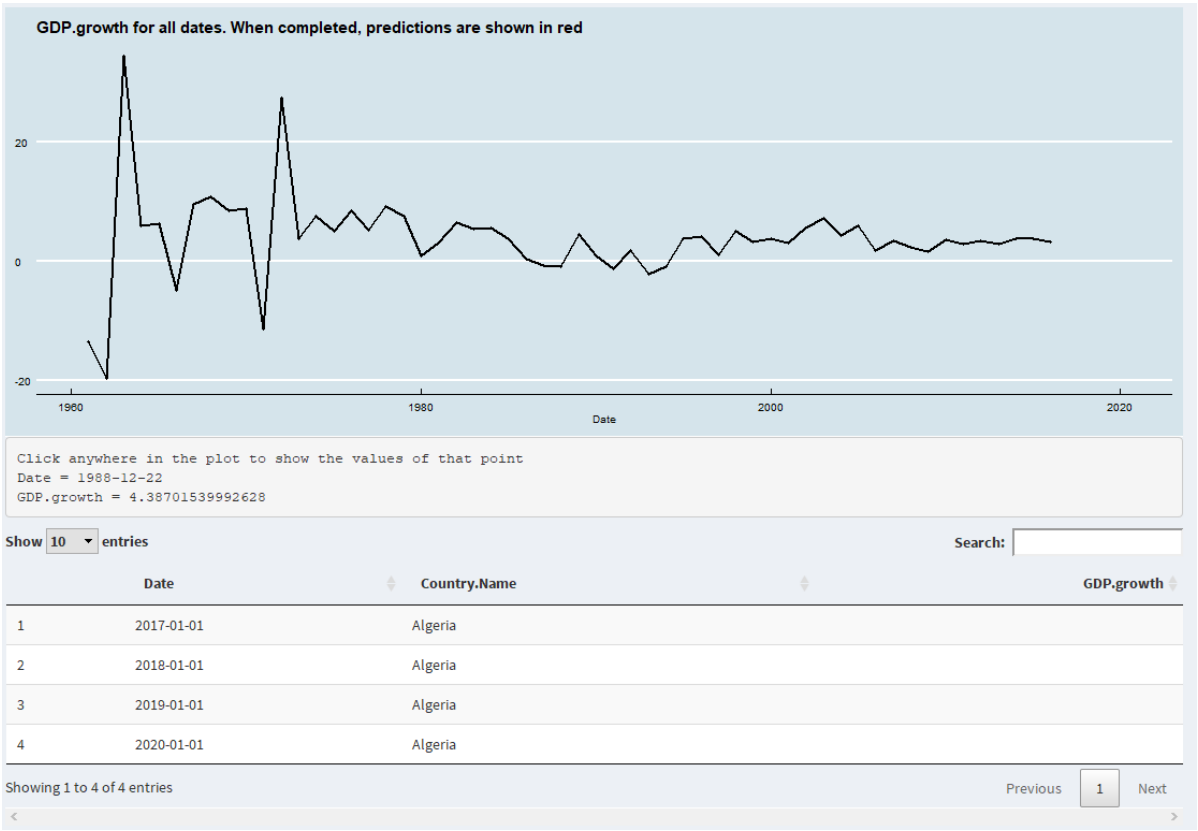


Figure 5-17 Time series analysis tab main body before predictions

The graph shows all actual observations for the variable the user has chosen to predict, and by clicking on any point in the graph, the field below will display the exact values for that point. The table below is like the previous tables; only now it is restricted to show the date, the user-defined constant variables, the predictions, and variable to be predicted.

To choose a different maximum run time per prediction period, the user can make a selection from the dropdown menu; options are 5/10/15/30 minutes and 1/2/3/4 hours. Once the machine learning is initialized, a small progress notification will appear in the bottom right corner of the screen, as shown in Figure 5-18.

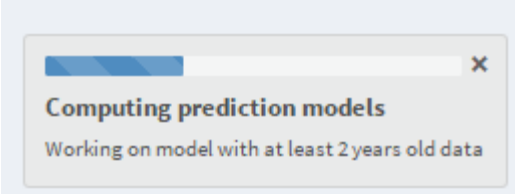


Figure 5-18 Time series analysis progress notification

The blue line pulsates, which helps show that the program has not frozen, but the line only increments after each prediction model. Once the notification disappears, then the models are

finished, and the user can press the Confirm button to see the predictions, like in Figure 5-19.

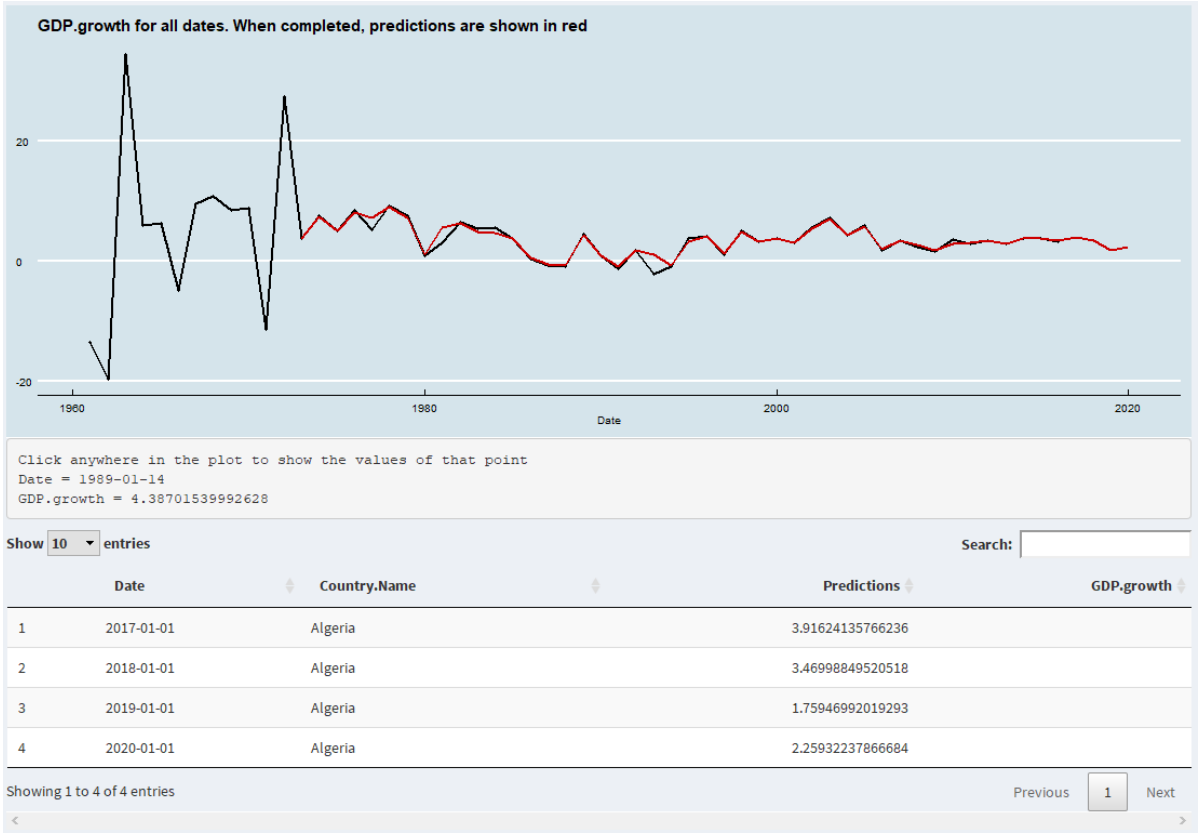


Figure 5-19 Time series analysis tab main body with predictions

The predictions are shown as far back as they go in the graph to give the user an immediate impression of how accurate they are. Because the models are tested on a subset that is not included in the training set and because of H2O autoML's measures against overfitting, the accuracy of predictions against observed values should be a good indication of the model's quality. Under the assumption that models with access to more recent information should be better, the combined predictions are simply those of the most recent model that can predict that time period. The download button in the time series analysis tab downloads the entire dataset with the combined predictions as well as the predictions of each model but otherwise works just like the previous download buttons.

While this tab is quite simple when it comes to the user interface, the functions that generate predictions are much more complex. The Confirm button works like in the previous tabs, in that it updates the dataset with new values, which includes the new values generated by the machine learning. However, the Confirm button also updates the values in the Evaluation tab.

The drop-down menu for selecting how long to run each machine learning model works like the function selection menus in previous tabs, allowing the user to choose between 5 minutes and 4 hours with natural increments. The user's selection, or the one-hour default, is translated into an integer with the corresponding number of seconds, which is used as an input in the H2O machine learning functions. I chose to let the users select the run time for each model because that hopefully increases their tolerance for long run-times and allows them to run shorter tests.

Clicking "Begin machine learning and model prediction" initializes the process that everything else in the application has prepared for. First, the prepared data is saved both as a working dataset for being fed into the models and as a presentation dataset for showing the results. This split is necessary because each of the models work iteratively, and if the results of the first one was saved into the basis for the second, then it would ruin the process. The presentation dataset then gets an extra blank column called "Predictions," which will hold the combined results of all the prediction models.

Variables that should not be lagged are defined as the constant variables, the date variable, and the other time variables, and these character vectors are combined into a single vector. All the other variables are lagged with **dplyr**'s `lag` and `mutate_at` functions, which create the required lags and names them a combination of the original variable name and an underscore followed by the lag number in three digits with leading zeros. For example, if a variable is named "Population growth" and the functions need lags from 2 to 10 previous periods then 9 new variables will be created, with the first named "Population growth_002". When viewed as a single row next to the original variable, the values in the lag variables correspond to the value of the original variable, only x rows above, where x is the lag number for each variable.

Further, all time variables other than the date variable are removed from the working dataset to avoid overlap with a later function, which expands the date variable to all possible time definitions. Then all the lag numbers of the different variables are extracted in an integer vector, and the variables without lag numbers are assumed to be lag 0. A new blank dataset is also created to hold the evaluation metrics for each model.

The **H2O** functions all work inside a `for` loop and this `for` loop is nested inside **shiny**'s `withProgress` function, which generates the progress notification window. The `for` loop runs for `i` in a sequence from one plus the gap to the gap plus the number of prediction periods.

Thus, `i` also sets how recent data each model can use. The first action in each loop is to increase the progress bar by one divided by the number of prediction periods; this will sum to 1 with the final loop and is the only progress updating.

The relevant data for each model, with actual observations, is extracted by comparing `i` to the vector of lag numbers and only including variables with a lag number of at least `i`. Variables that should not be lagged and the prediction variable are included as well. Then the `tk_augment_timeseries_signature` function from the **timetk** package is applied to add all possible time period specifications to the data as new variables, which should help account for all strictly time-related effects, such as seasonality. (Dancho & Vaughan, 2018)

Character and ordered data are transformed into factors because the **H2O** functions can't utilize those forms of data, and then the relevant data for each model is saved as a special **H2O** object. That **H2O** object is then split into 80 and 20% partitions randomly, using **H2O**'s `h2o.splitFrame` function. The 80% partition is saved as the training set, while the rest is then saved as the test set, which will be used to evaluate the performance of each model. I use an 80% split because **H2O** automatically creates a validation set from within the training set and the 80% limit was shown as the standard in H2O's documentation. (H2O.ai, 2019)

The prediction dataset for each model is set as the generated future time periods that are less than or equal to `i` periods into the future. That is because each model can only predict a maximum of `i` future periods. Otherwise, the prediction dataset is processed just like the relevant base data and saved as an **H2O** object. With all of those steps completed, the `autoML` function can finally be started. The **H2O** `autoML` function is full of intelligent default settings that automatically adapt to the data and therefore, as little as possible is set by the user. The actual function then only needs the user's chosen prediction variable, the training dataset, the test dataset, and the maximum number of seconds it can run. By only defining which variable should be predicted, all other variables are used as predictors. Specifying a training set and a test set enables the function to use the automatic cross-validation as well as the test set to prevent overfitting (H2O.ai, 2019). The machine learning uses all the computation power allocated to it, and if it is running on a local computer, it will therefore usually not be practical to use the computer for anything else while it is working.

Each attempted prediction model is stored in an **H2O** environment object, and when it is finished, the performance metrics of the best model is extracted with the

`h2o.performance` function. Then the most essential variables for the best model are extracted and saved in a separate table. The predictions are saved in a column of the same length as the extended base dataset by adding missing values to the beginning and end to fill in for missing predictions. Then that column is added to the presentation dataset that was prepared previously, with its own name. Under the assumption that the model with access to the most recent data should be best, the combined predictions are set as the first prediction model, with the missing predictions at the end filled in by the last prediction of each subsequent model. Finally, the corresponding row of the pre-prepared evaluation metrics table is filled with values extracted from the performance object that was saved previously. Then the value of `i` increases by one, and the loop begins again.

After the `for` loop is finished and models have been created for all prediction periods, the presentation dataset is saved so that it will be rendered to the user when they press Confirm and a complete evaluation metrics table is saved as a global object so that the user can see it in the Evaluation tab.

The last button in the sidebar is another Download as .csv button, it works like previously, downloading the entire dataset including all predictions. There is a graph and a small table in the main body of the Time series analysis tab. The graph is created with the **ggplot2** package and shows the date along the x-axis and the variable to be predicted along the y-axis (Wickham et al., 2019). When the machine learning is completed, and the user clicks Confirm, the combined predictions are shown in the same graph. The color scheme in the graph is from the **ggthemes** package (Arnold et al., 2019). Since the variable to be predicted can cover any number of ranges, and it is often difficult to read precise values from the y-axis, I chose to remove the y-axis entirely and instead let the user click at any point in the graph and display the exact values of that point in a field below the graph. Removing the y-axis also frees up space to make a more streamlined graph. This click functionality is added with standard functions from **shiny** by specifying that the x and y coordinates of any click inside the graph should be saved as an input. The x and y values can then be rendered as text to the user and update immediately due to the reactive nature of shiny.

The table at the bottom of the main body only shows the generated future time periods, and then only the date, user-defined constant variables, predictions, and the variable to be predicted. Naturally, the variable to be predicted has no observed values in the generated future, but it included to clarify which variable is being predicted. The table is rendered as a

reactive shadow with the aforementioned restriction so that it is not a separate dataset. I chose to limit the table because the user has had the chance to inspect all the data in both previous tabs and the precise actual future predictions are the new and most important data. The graph helps illustrate how accurate the predictions are, while the table presents concise results.

5.2.4 The Evaluation tab

The last UI page in the application is the evaluation tab, which provides extra information meant to give the users more insight into how the machine learning has worked and how well each model has performed. There is only one user input here, a dropdown menu connected to the data that allows the user to choose which model to evaluate and the rest of the tab is packed with information as shown in Figure 5-20.

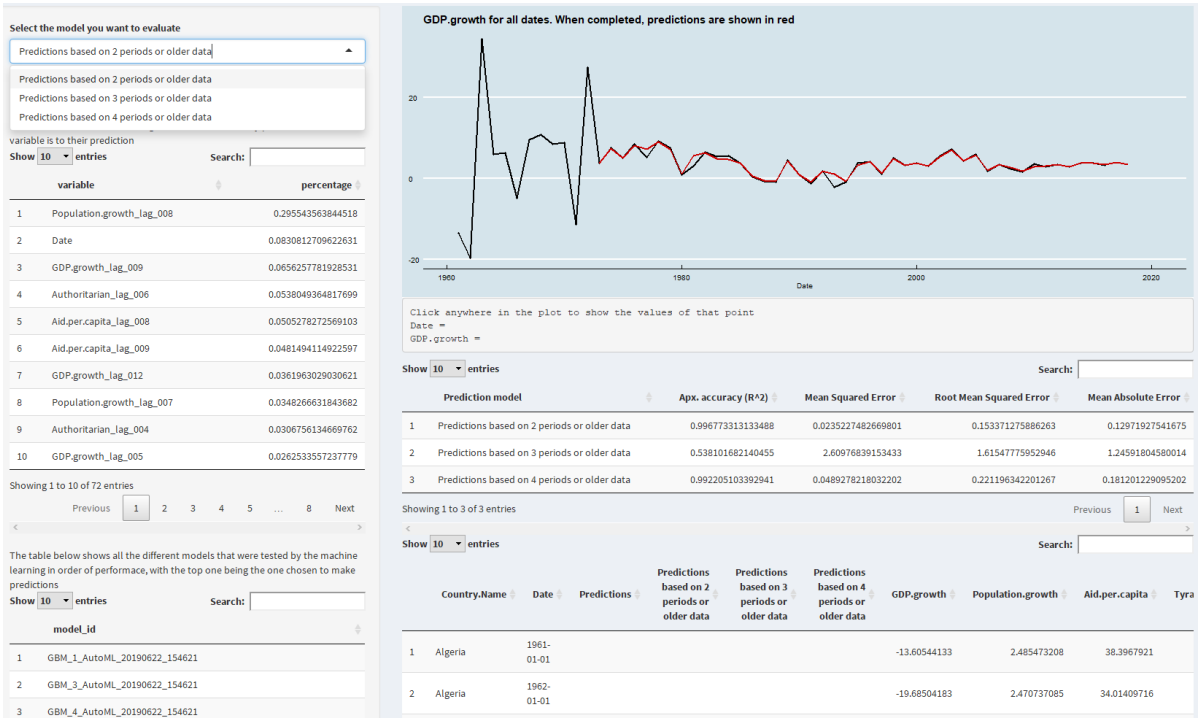


Figure 5-20 The Evaluation tab

Below the model selection menu in the sidebar, there are two tables. The first shows the variables used for making predictions and their percentage importance to the chosen model, while the second shows a ranked list of the different machine learning techniques that were attempted by the AutoML algorithm with the first one as the chosen leader. The tables work in the same way as the previous ones. In the main body, there is the same graph as in the

previous tab, except that it only shows predictions from the model chosen by the user. Underneath the graph, there is a table with some key statistics for each model: R^2, Mean Squared Error, Root Mean Squared Error, and Mean Absolute error. At the bottom, there is another table containing all the data along with predictions from each model, which is the same data the user can download in the previous tab.

The evaluation tab is more technical than the previous tabs, and as such, it is more suited for users with some experience in data analytics. Sadly, the way each **autoML** model works can be mysterious, and the variables that are ranked highly should not be interpreted as being causally important for the outcomes of the predicted variable. A key advantage to the variable rankings is that if a variable seems to have no importance in any of the models, then the user could potentially save time in the future by not gathering data on that variable again.

The dropdown menu in the Evaluation tab is actually a variable selection input, just like in the first tabs, except that it links to a table that is not displayed to the user. The remaining tables are shown based on the number in each variable name, which corresponds to the `i` value of each prediction model from the for loop, and the number is extracted with the `parse_number` function from the **readr** package (Wickham et al., 2018). Other than that, all the remaining tables are just rendered in the same way as before, using the **DT** package, and refer to reactive shadows of global objects that were created as part of the machine learning process (Xie et al., 2019).

6. Comparisons with other interfaces

The accuracy and quality of any predictions from my application are almost entirely based on the user's data and the AutoML algorithm, which is created by H2O.ai. Therefore, using prediction results for advocating my application is not particularly relevant[8]. My contribution lies in the automation and simplification of the processes that are necessary for applying AutoML to get time series predictions and the data preparation capabilities that made up the majority of my efforts. Therefore, the most relevant comparisons are of the design and extra functionality of alternative automatic machine learning frameworks. Comparisons of design are naturally subjective, which is why I will document the other interfaces in detail and include screenshots to avoid making subjective descriptions. Comparisons of functionality are objective in the cases where one application can do something another cannot, but when both can do the same thing in different ways, it becomes subjective once more. Since I am an expert in my application and not as proficient with the alternatives, I will give the other interfaces the benefit of the doubt in cases where I do not manage to document their advanced functionality properly.

Such comparisons are relatively extensive, and so I will only present two alternatives, one open-source that is freely available to those who could use my application, which my application needs to beat in one or more ways to be a viable contribution, and the highest-rated monetized service that advertises the same functionality as my application. The open-source comparison is with H2O's Flow UI, which is installed along with the extension that enables AutoML through R, and I chose it because it is an immediately available alternative to my application. The monetized comparison is with RapidMiner Studio Professional because RapidMiner is one of the highest-rated providers of analytical tools and their Professional version adds their Turbo Prep and Auto Model extensions, which are advertised to perform almost exactly what my application is designed to do, only not focused on time series predictions. Naturally, I do not expect my application to be as good as a product made by a company of experienced professionals, but since their base Professional license is priced at $

[8] Evaluations of the accuracy and efficiency of the H2O AutoML algorithm can be found in papers such as "Benchmarking Automatic Machine Learning Frameworks" by Balaji, Adithya and Alexander Allen (2018) and articles like "AutoML — A Tool to Improve Your Workflow (Updated)" by Tom Allport (2019), which I refer to in the presentation of AutoML.

5,500 per user, per year, my application can be an alternative for more price-sensitive users and realize some relative benefits from its specialization on time series predictions.

6.1 Comparison with H2O Flow UI

Anyone with a local version of my interface can choose to use Flow UI as well, simply by entering localhost:54321 in their internet browser address bar. Naturally, the Automatic Machine Learning algorithm is exactly the same, but the design and data manipulation functionality is quite different. Flow UI's initial display is shown in Figure 6-1, which is in a Google Chrome browser.

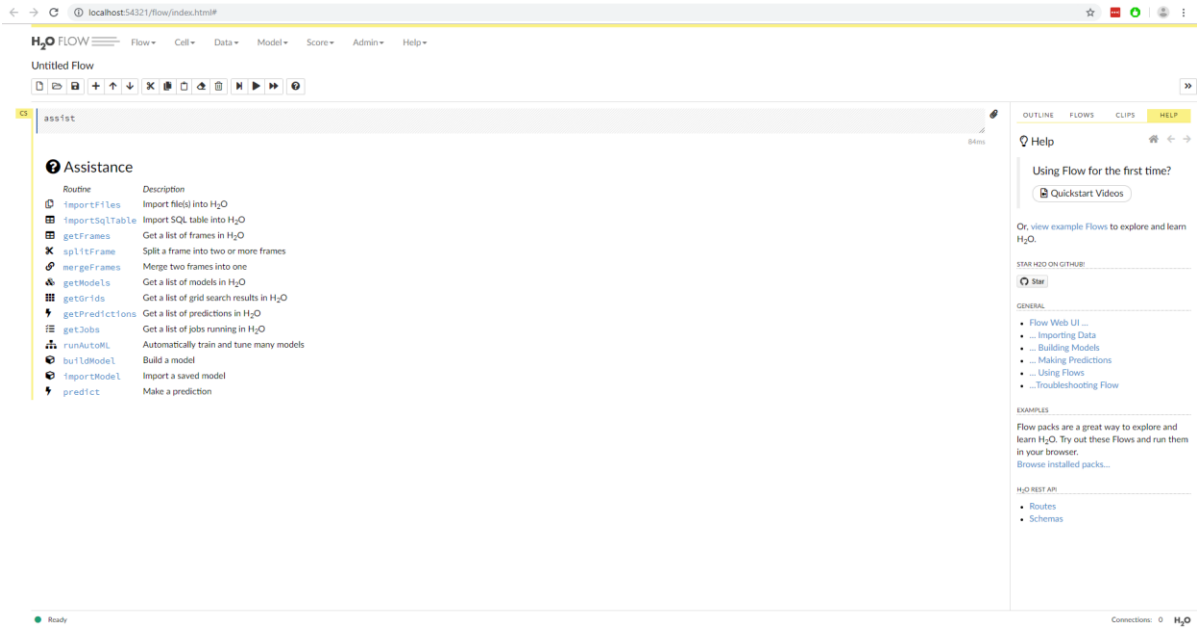


Figure 6-1 The initial interface for H2O's flow UI

The first and most obvious difference to my interface is that Flow UI has many more possible user interactions, which is quite natural, as it is a general tool and my interface is entirely focused on data preparation and time series analysis. With the assumption that the user is interested in making time series predictions into the unobserved future, the comparison will follow the typical workflow in my application.

To upload data to Flow UI, the user can click the importFiles link in the overview, which makes a new prompt appear in the bottom of the display as shown in Figure 6-2. This is how all of Flow UI's interactions work, a function is selected, and a new prompt appears in the

bottom of the main display. The interaction is like a mix between a normal programming window and a GUI.

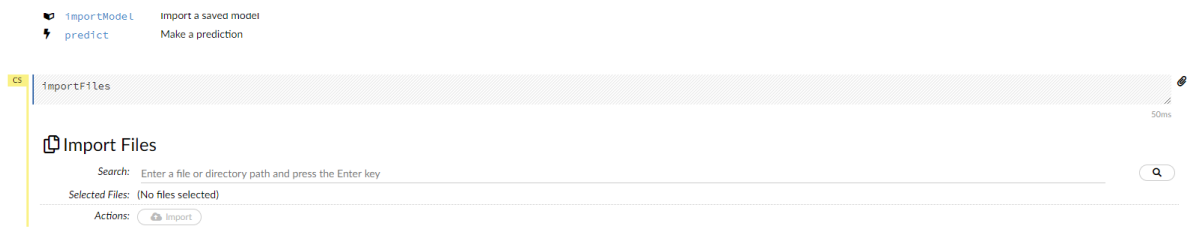


Figure 6-2 Flow UI's initial file import prompt

To import a file, the user must type in or copy a file path, click the search icon, select the desired file from the search result and click the import icon. An example of this is shown in Figure 6-3.

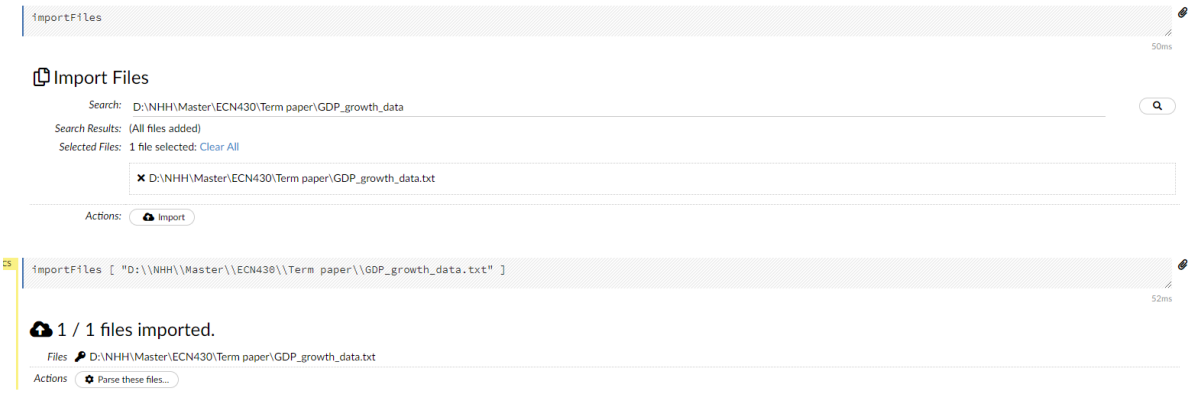


Figure 6-3 Flow UI import file function with selected file

Once a file is imported, it must be parsed to be read into Flow UI as data. The parse setup appears once the user clicks the "Parse these files" button and is shown in Figure 6-4. This setup is relatively similar to in my application, but has some additional options and most notably, the ability to override the detected data type of each column. The file is processed once the user clicks "Parse" at the bottom of the prompt, and once it has loaded, the user can click view to show summary statistics for each column.

Figure 6-4 Flow UI file parse setup

Once a dataset is parsed it is stored as a "Frame" in the H2O Cloud and available for model building. However, it is not possible to edit the data once it is uploaded. Flow UI has functions to impute missing values and merge datasets, but if the data is in the different formats, such as wide and long, Flow UI cannot merge the data. In fact, the data must be in the long format, with one variable per column and different observations in each row for Flow UI to use it all. There is no way to transpose the data inside Flow UI. If all the datasets are correctly formatted and ready for analysis before they are uploaded, they can be merged into a single dataset, using the function shown in Figure 6-5.

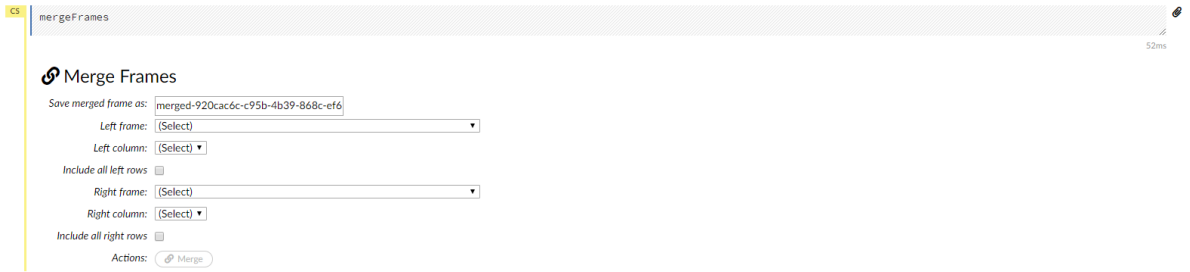


Figure 6-5 Flow UI merge datasets function

The merge function can only use one identifying variable, so basing it on a name and a time period is not possible, and this identifying variable must also be created before uploading. Imputing missing values can be completed for one column at a time. However, only the simplest methods are available; mean, median, and mode, as shown in Figure 6-6.

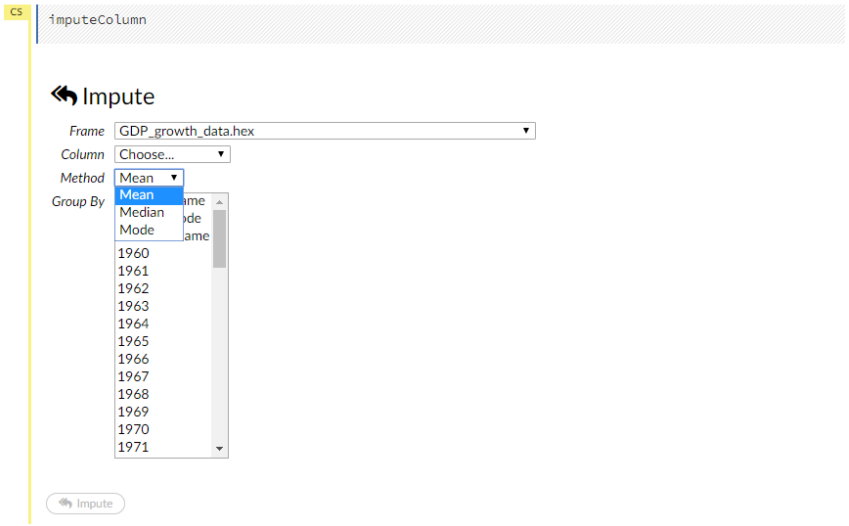


Figure 6-6 Flow UI impute function

Once all the data is ready, the user must split it into appropriate subsets, using the split frame function. The user can split the data into as many random subsets as desired, specifying the ratio for each split, as shown in Figure 6-7.

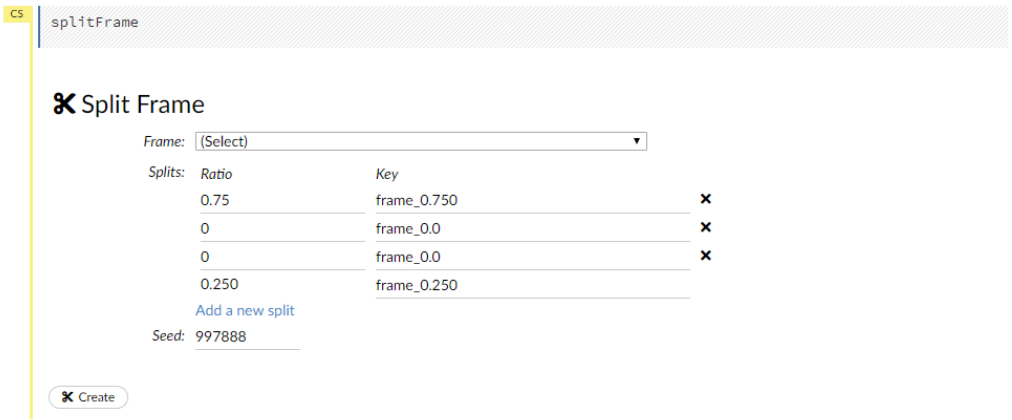


Figure 6-7 Flow UI split frame function

After splitting the frames, everything is ready for building the machine learning models. This is accomplished with the runAutoML function, which is the same as in my application, but in Flow UI the user can make their own decisions with regards to all the options that are left automated or handled in my application, as shown in Figure 6-8. Many of those settings require some technical knowledge to understand, and therefore they might confuse novice users, especially since it is not clear which inputs are required and which can be left blank.

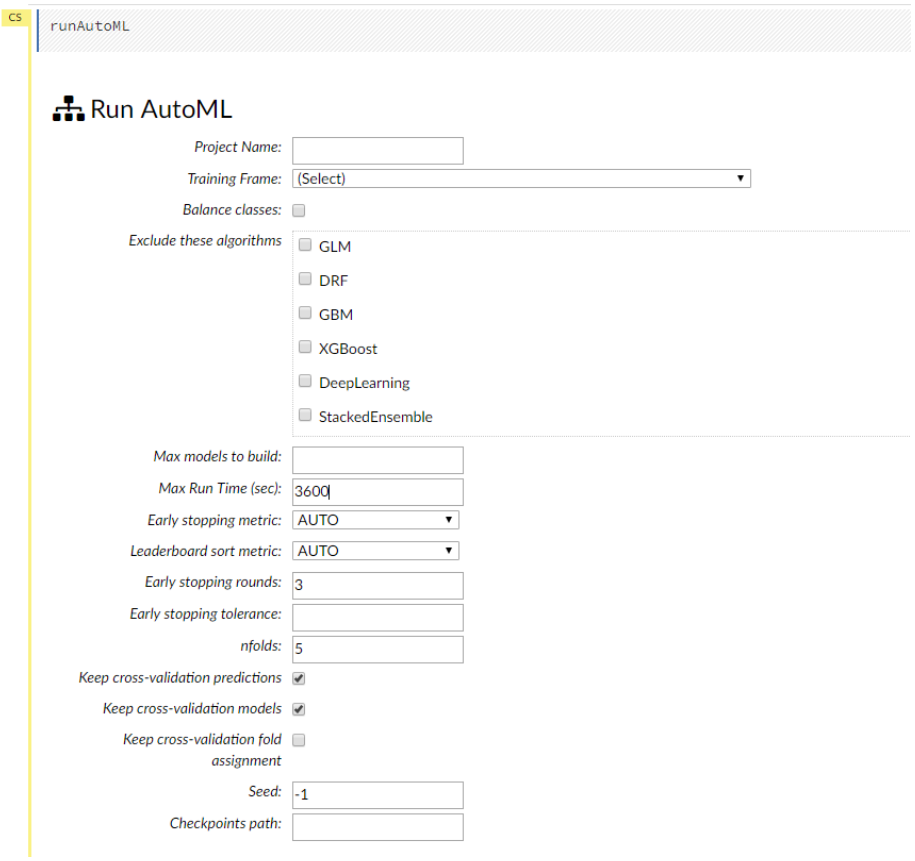


Figure 6-8 Flow UI AutoML function

Once the models are completed, the user must look at the leaderboard to find the best model, so that it can be used to make predictions. The leaderboard is found by clicking on the View icon in the Job prompt; then the user can click on the top model name in the leaderboard to display that model, as shown in Figure 6-9.

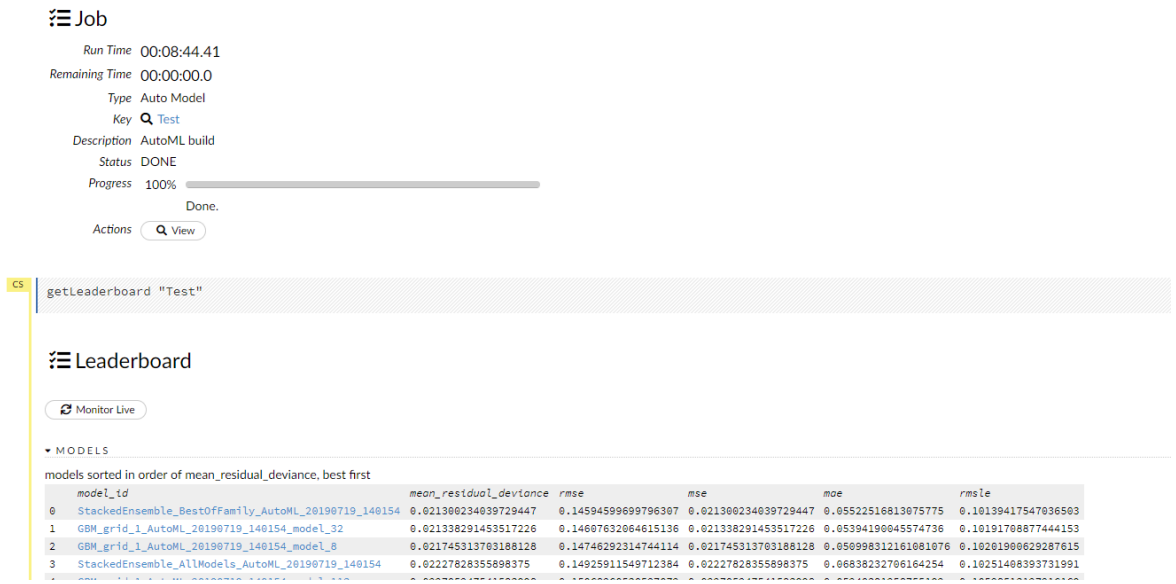


Figure 6-9 Finding the correct prediction model after AutoML in Flow UI

The model overview is shown in Figure 6-10 and serves as a navigation menu for all the different applications of the model. Users can look at performance metrics and all the technical information, but there are no visualizations such as graphs. To use the model for predictions, users can click the Predict icon, which leads to the prompt shown in Figure 6-11.

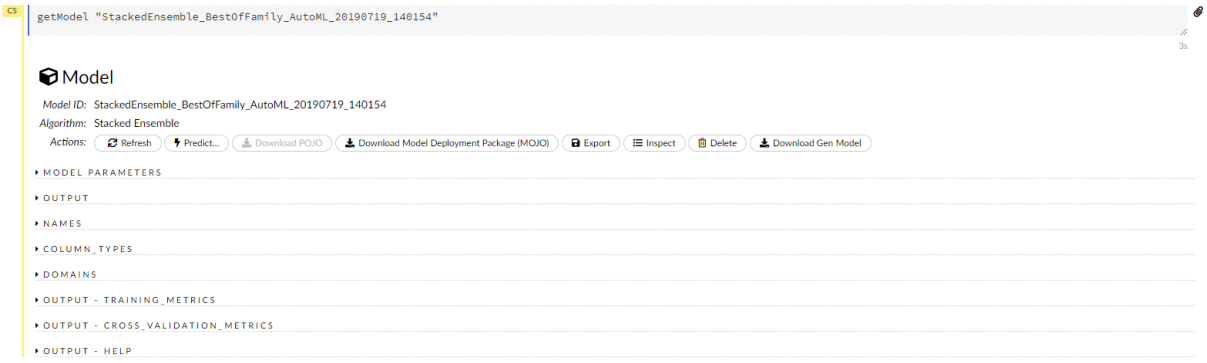


Figure 6-10 Flow UI AutoML Model overview

However, the predict function only accepts a selected model and a dataset as inputs. The dataset must contain the same variables as those used in training the model, and predictions are made for each row. This system means that for users to actually get predictions of the unobserved future, they must either provide a dataset with expected values for all variables, which is a comprehensive analysis on its own, or use datasets that have all the relevant historical data on each row throughout the entire process. Further, everything would have to be completed once for every period into the unobserved future the user wanted to get predictions for and combined manually.

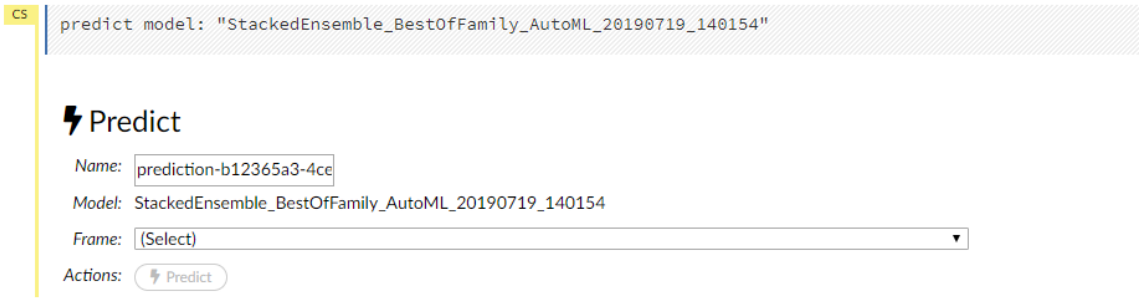


Figure 6-11 Flow UI predict function

6.2 Summary of comparison with Flow UI

In short, Flow UI has almost none of the added functionality in my model[9], relying on extensive preparation in another program before it can produce the same results. The uploading of data is similar in both interfaces, with Flow UI having many more steps and options, which is good for experienced users that might be constrained in my application but could be confusing for novice users. When it comes to data preparation, Flow UI only offers functions that impute missing values and merge datasets, but these are so simple that they could provide limited value. Other than that, Flow UI does not let the user edit or transform uploaded data, which is the main functionality in the first half of my application. Given a dataset, Flow UI has no integrated way to generate predictions outside of that dataset, which is the functionality added by my application in the second half. Therefore, Flow UI is objectively unsuited for making the sort of time series predictions my application is designed for, even though they are based on exactly the same algorithm. H2O Driverless AI, H2O's monetized alternative, could be an entirely different matter, and while it has a free trial, I compare with RapidMiner instead because it is higher rated and seemingly more similar to my application.

[9] Flow UI has many functions that my application does not support, but in the restricted focus on data preparation and making future time series predictions, my application offers more functionality.

6.3 Comparison with RapidMiner Studio Professional

The free trial of RapidMiner Studio can be downloaded from https://rapidminer.com/get-started/ by registering with the company. This comparison will focus on the uploading and preparation of data with the Turbo Prep extension and the creation of time series predictions with Auto Model. The initial display with Turbo Prep is shown in Figure 6-12.

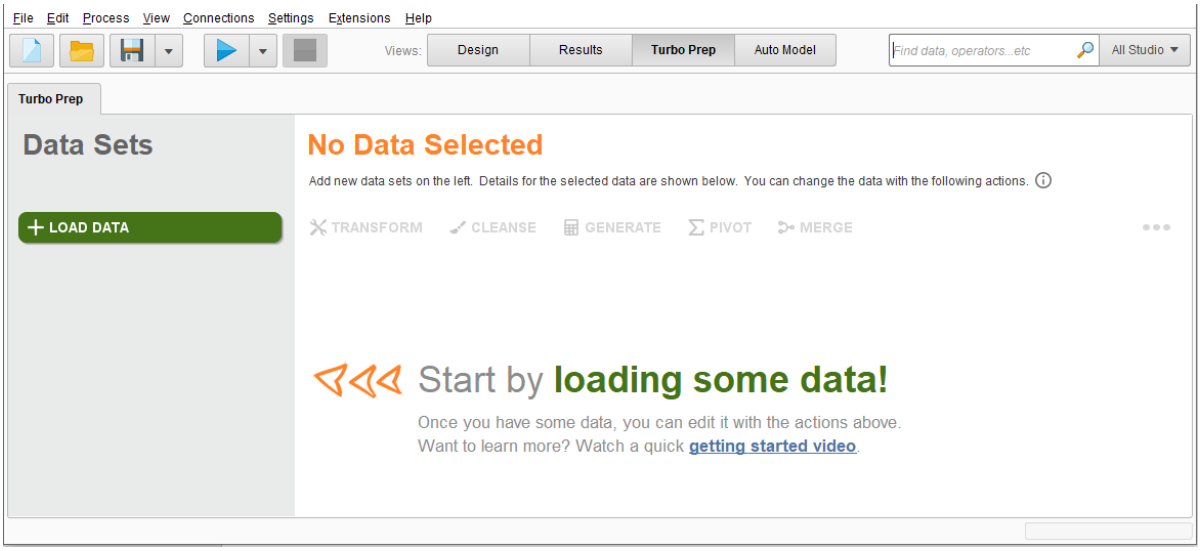


Figure 6-12 RapidMiner Studio initial Turbo Prep display

The Turbo Prep window seems very clean and streamlined, there are general options along the top of the window, but the Turbo Prep extension itself looks more intuitive than my application at first glance. Clicking on the Load Data button brings up the menu in Figure 6-13.

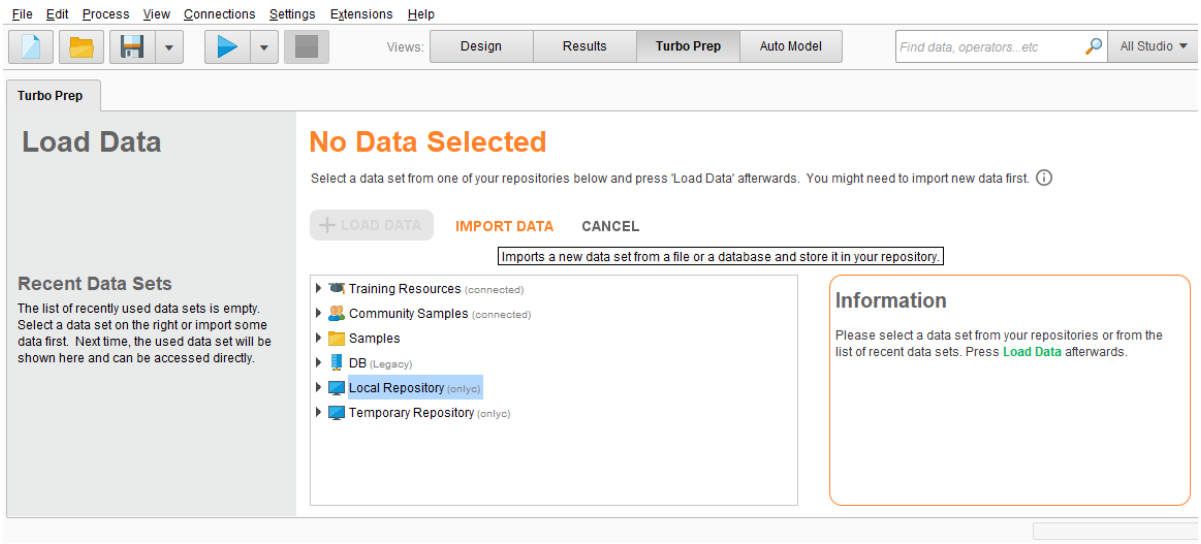


Figure 6-13 RapidMiner Turbo Prep Load Data

To load new data into RapidMiner Studio, the user must click on “IMPORT DATA” which brings up the prompt in Figure 6-14. Again, the interface is clean and informative and has support for connecting to databases, something my application does not. There is also a cancel button in the bottom right corner of the prompt.

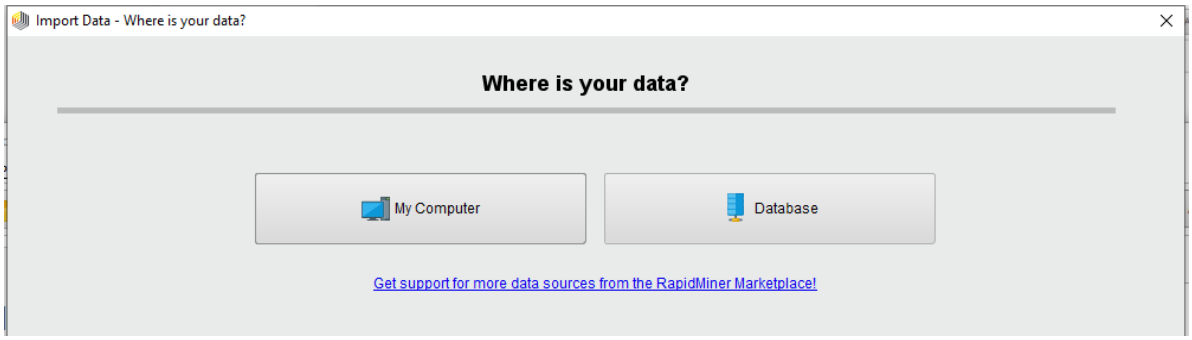


Figure 6-14 Rapid Miner Turbo Prep Import Data prompt

Clicking on “My Computer” brings up a relatively standard file navigation GUI, but still within the RapidMiner Studio Program. The display is shown in Figure 6-15 and when the user selects a file the “Please select a file to import” message above the cancel button turns into “The selected file will be imported as a: X Change” Where X is an automatically detected data type, and Change is a link to a mini menu to override the automatic selection.

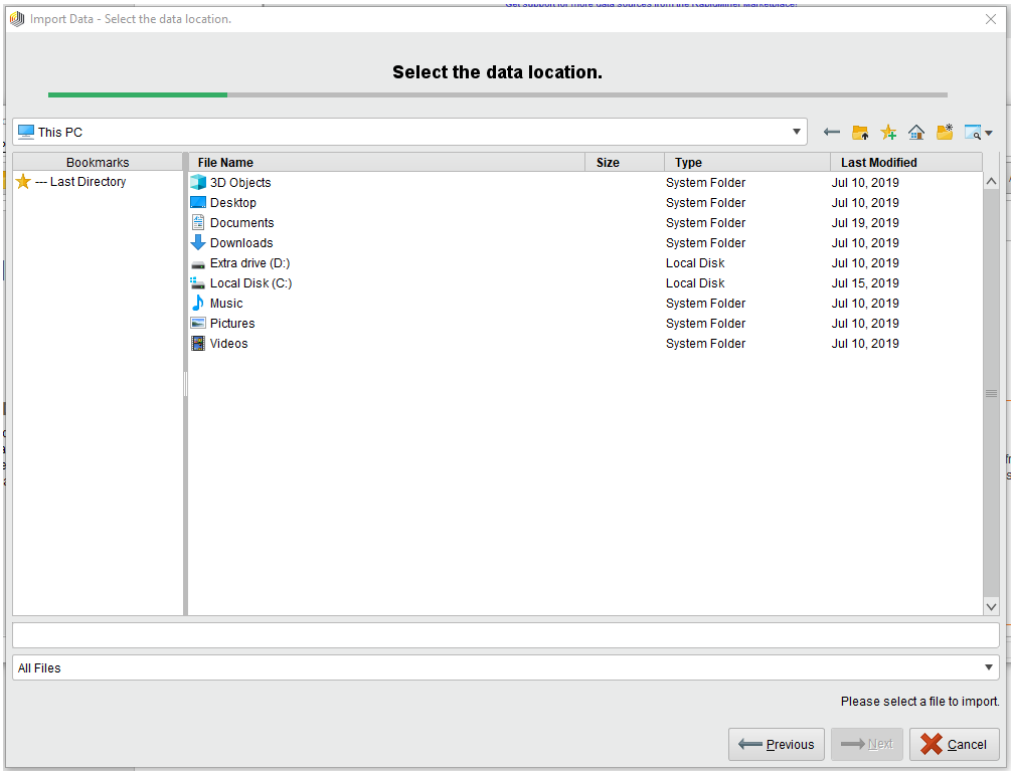


Figure 6-15 RapidMiner Studio Import Data file navigation

Clicking “Next” with a selected file brings up a preview of the data after it has been loaded, with automatic detection of the delimiter as well as other formatting options, as shown in Figure 6-16. The user can override the automatic formatting, unlike in my application, where everything is left up to the readr package (Wickham et al., 2018).

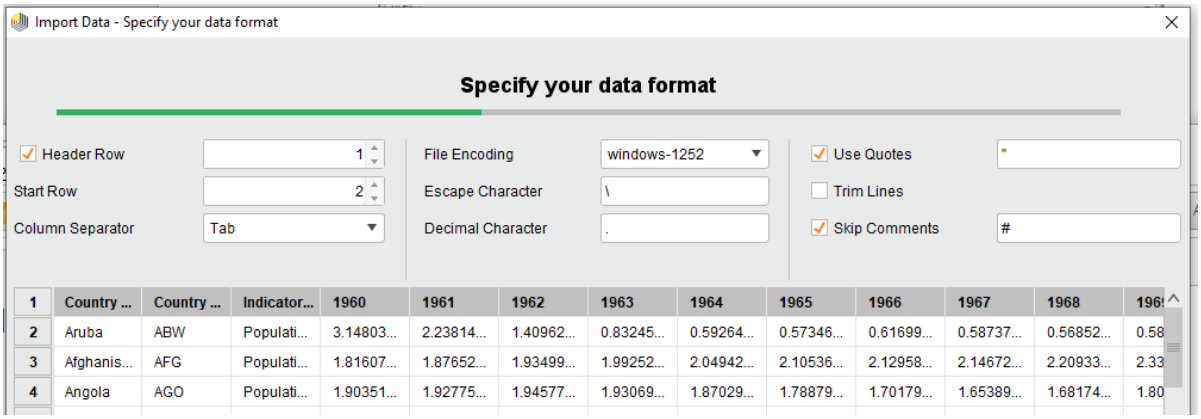


Figure 6-16 RapidMiner Import Data format specification

Clicking “Next” again brings the user to another formatting window, shown in Figure 6-17, where they can specify the formatting of individual columns and decide which data format to use.

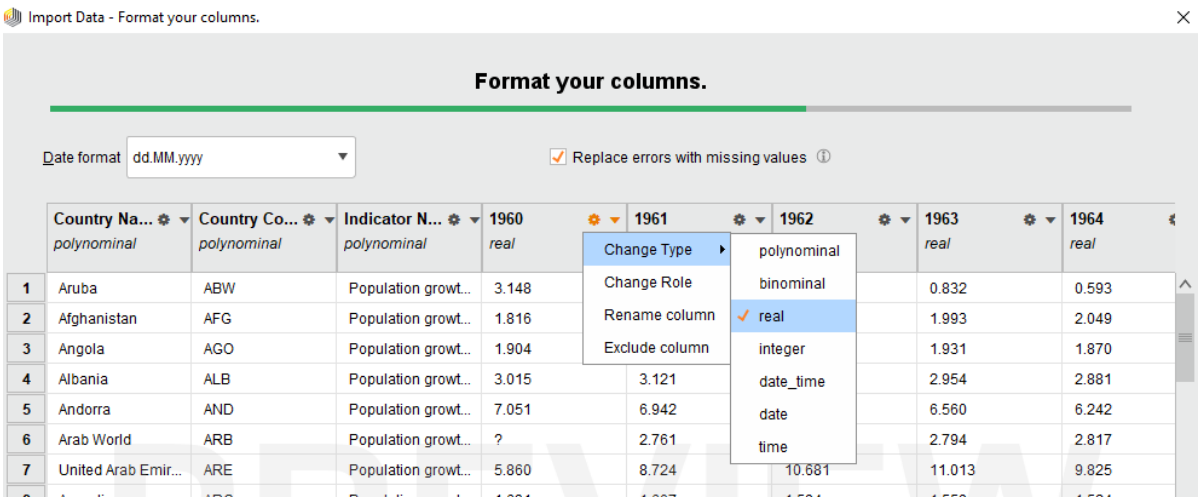


Figure 6-17 RapidMiner Import Data column format specification

Clicking “Next” one more time brings up the final section of the Import Data window, not shown, where the user selects where they want to save the data in the RapidMiner Studio system and then press “Finish.” After a loading screen, the Import Data window closes, and the user is back in the Turbo Prep section with the data loaded, as shown in Figure 6-18.

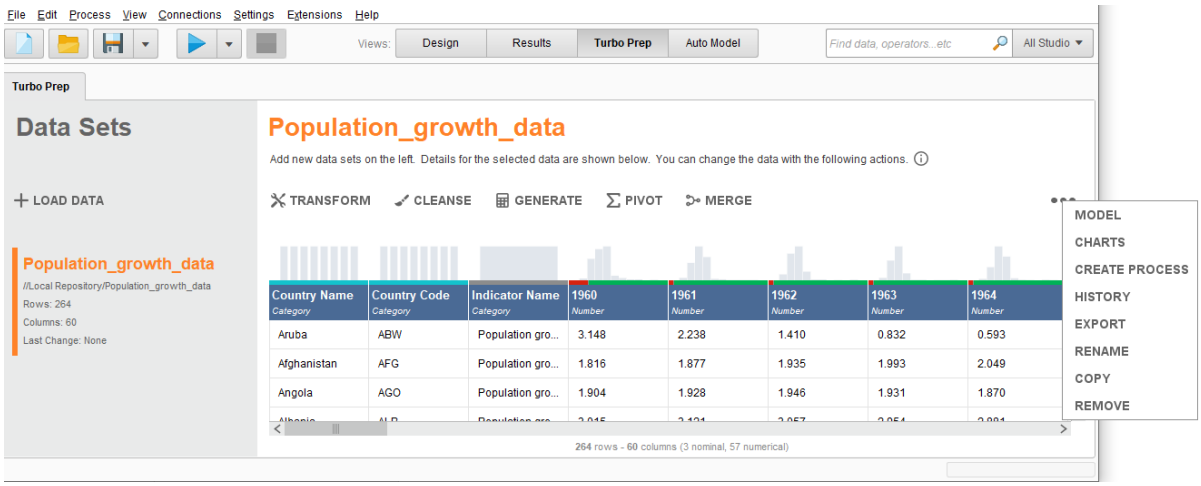


Figure 6-18 RapidMiner Turbo Prep with loaded data

With loaded data, the Turbo Prep section is full of information, including bars that indicate the number of missing values as well as a tiny histogram of the value distribution of each column. Clicking the “Transform” icon brings up the Transform subsection shown in Figure 6-19.

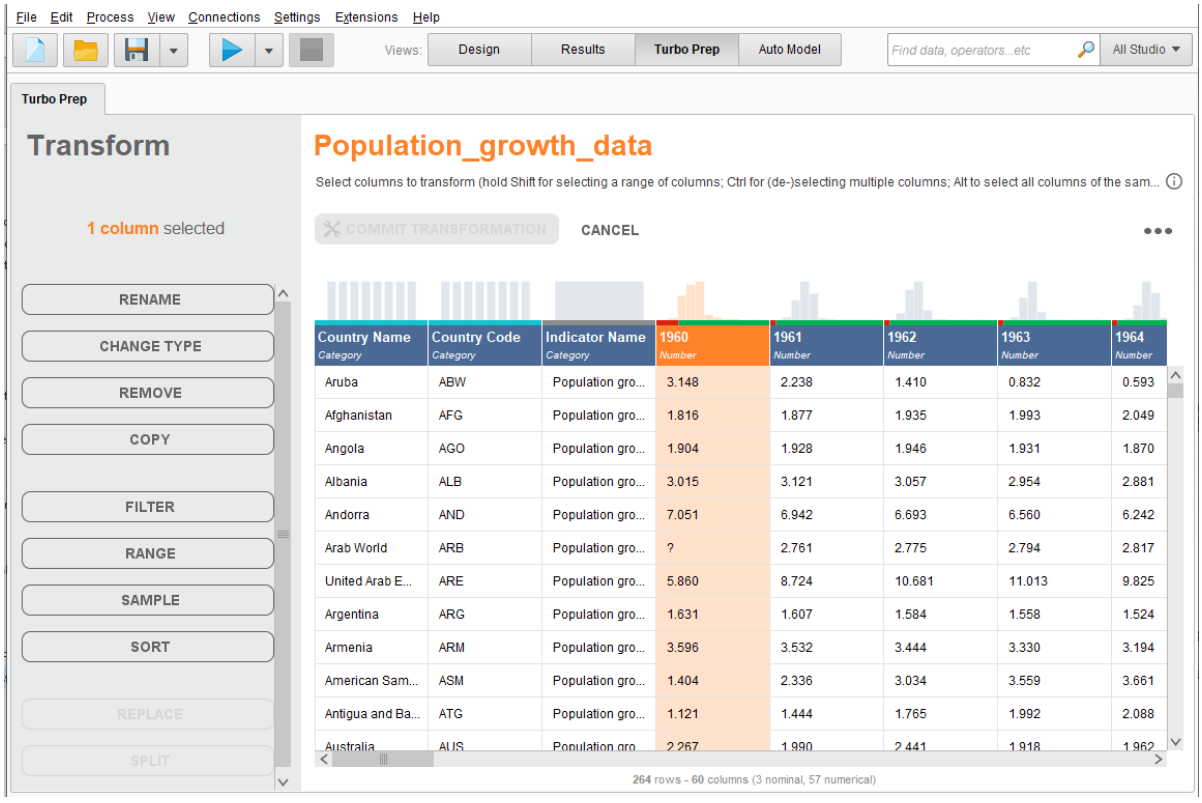


Figure 6-19 RapidMiner Turbo Prep Transform subsection

The different transformation options in the left column cover many of the functions in my application, while the options to change the type of a column, create a copy of a column with

a new name, subset the data based on row numbers, and reduce the table to a random sample are not in my application. The user can apply one of the transformations, see the result on the rendered table and then either click “Commit transformation” or “Cancel,” which is the same basic iterative change functionality as in my application. However, the “History” option in the extra menu in the far right of Figure 6-18 allows the user to roll the data back to before any of the committed transformations, which is a clear advantage. The “Cleanse” option in Figure 6-18 is meant to be a final preparation before machine learning models, and therefore, I will cover it last.

The “Generate” option brings the user to the subsection in Figure 6-20, in which the user can create a new column by defining a formula that can involve multiple other columns. My application only offers the ability to generate a new column with a single value for all observations. Because the automatic machine learning looks for all types of relationships in data and can do feature generation, performing intelligent transformations on the data is not as necessary as in normal data analysis, but it is very useful for manual feature engineering (RapidMiner, 2019). Again, the user can apply a generated column, see the preview and commit it if desired. However, there is no function for creating lags of variables in this section of RapidMiner Studio.

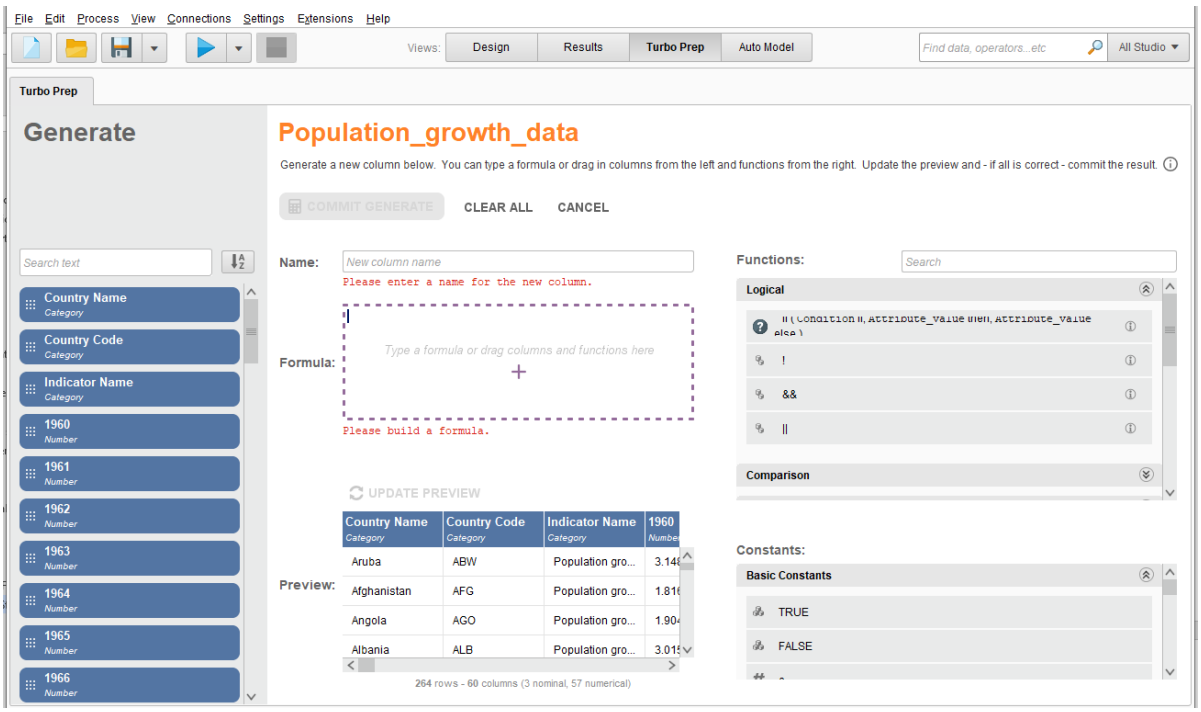


Figure 6-20 RapidMiner Turbo Prep Generate subsection

Turbo Prep also has a “Pivot” option, which is not really for preparing data, but for performing visual data analysis. The user can define categories and look at different aggregations in both a table and in graphs. While it is a useful tool, it is not relevant for the automatic machine learning and cannot be used to transpose data. To transpose the data, users must go outside of Turbo Prep and to the regular Studio interface, which has a completely different setup, shown in Figure 6-21.

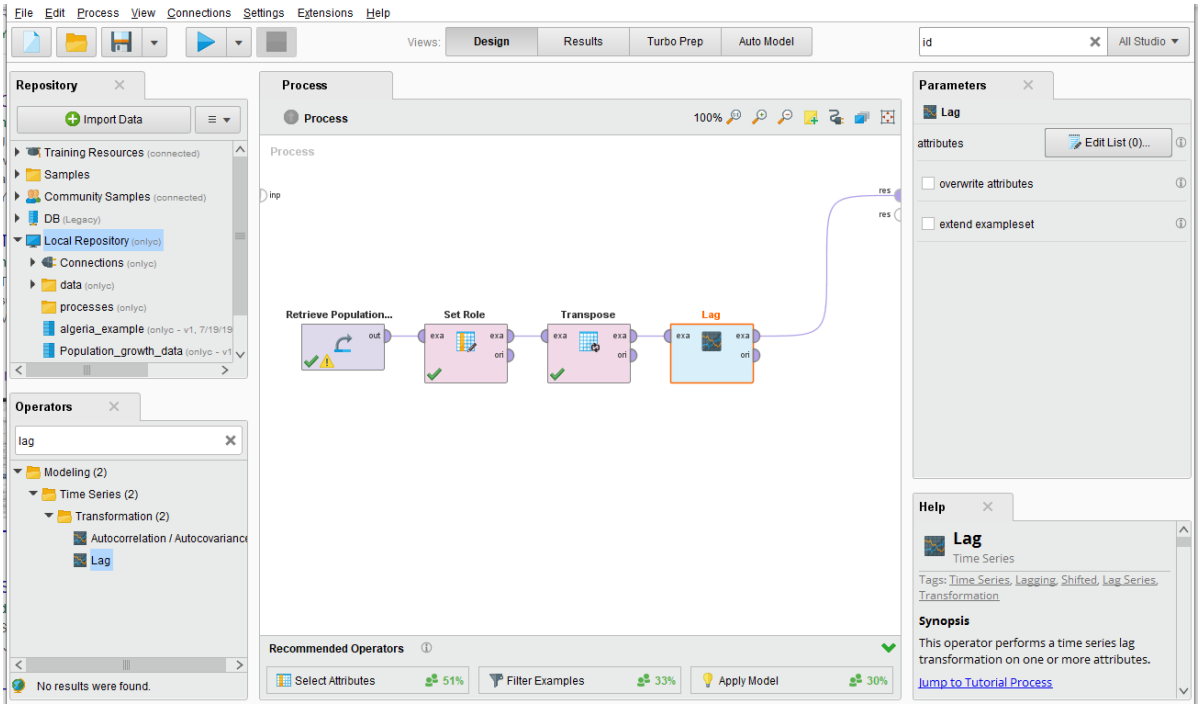


Figure 6-21 RapidMiner Studio process design

The regular studio system is less intuitive and very similar to programming, requiring users to spend time on learning the RapidMiner Studio system. Users must also use the regular interface to create lags of variables, which could be cumbersome, as it seems like every lag of each variable must be specified individually.

Going back to Turbo Prep after transposing data, the user can easily merge the data with the “Merge” option, shown in Figure 6-22. By allowing different types of merging as well as matching any number of identifiers that could have different names, the merge function is much more flexible than in my application.

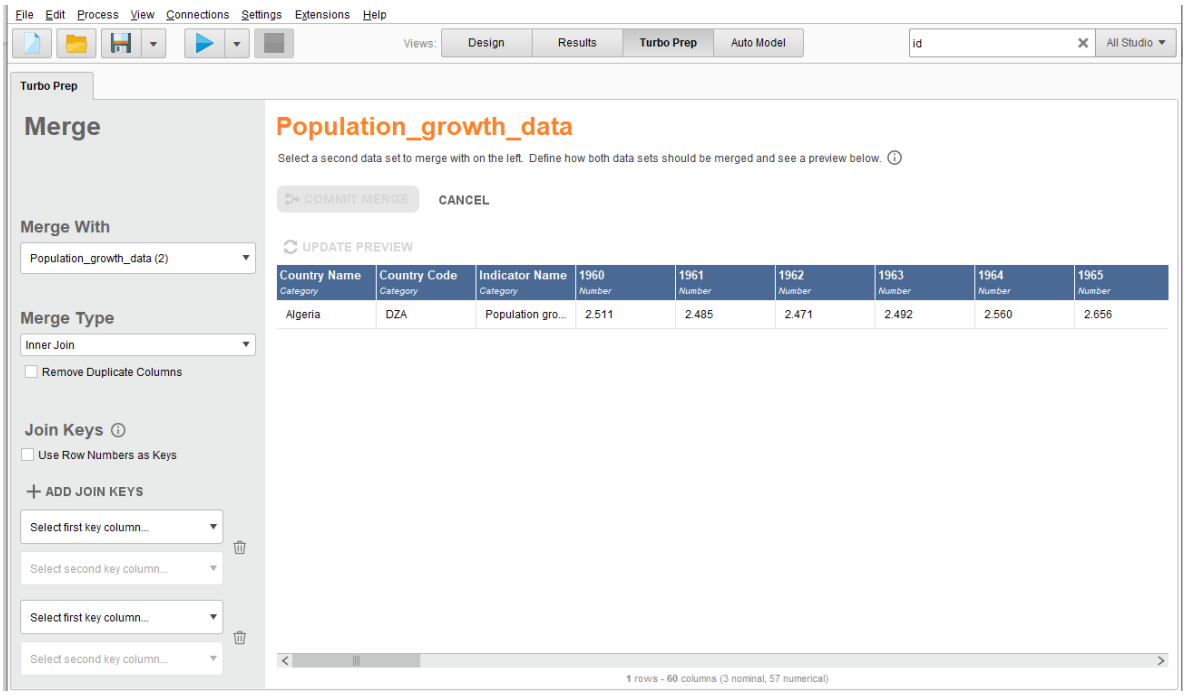


Figure 6-22 RapidMiner Turbo Prep Merge subsection

Once the user has uploaded and merged all their data, they can use the "Cleanse" option as a final preparation for Auto Model. The cleanse function allows the user to remove correlated columns, deal with missing values, and otherwise optimize the data for analysis, as shown in Figure 6-23. Such cleansing is not strictly necessary, but it can drastically reduce the required computation times.

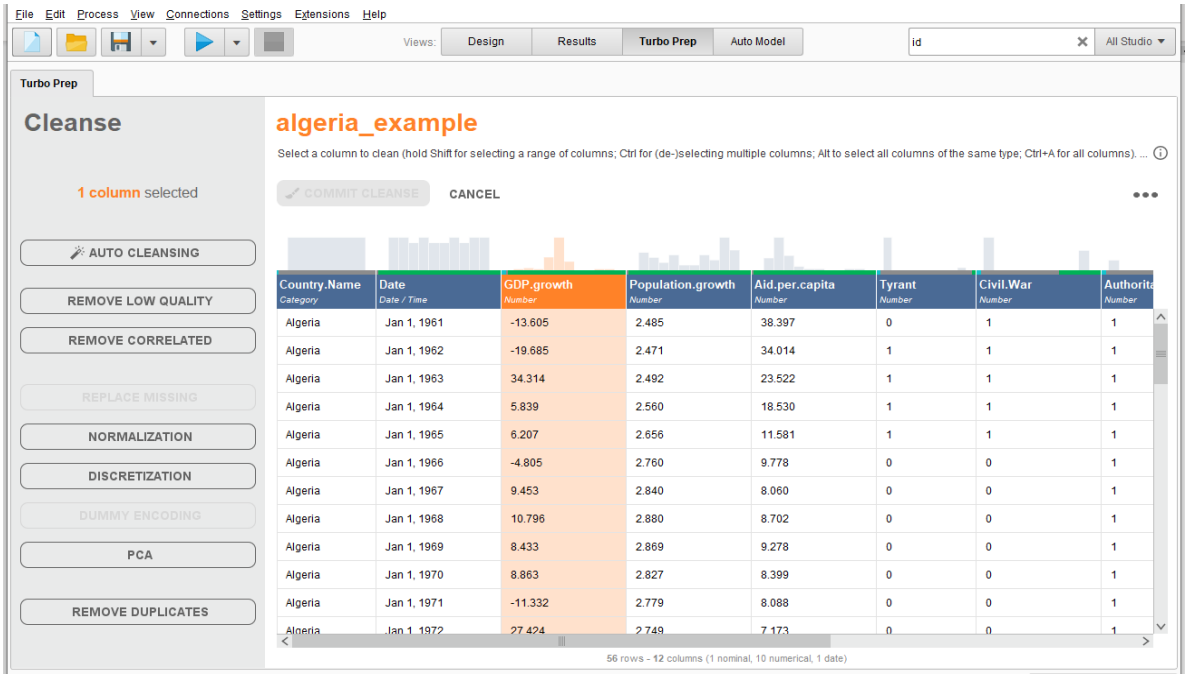


Figure 6-23 RapidMiner Turbo Prep Cleanse subsection

With all data preparation finished, the user can move to the Auto Model extension, shown in Figure 6-24. Auto Model requires at least 100 rows of data, which could be a problem, especially with yearly data, and is not a limitation in my application.

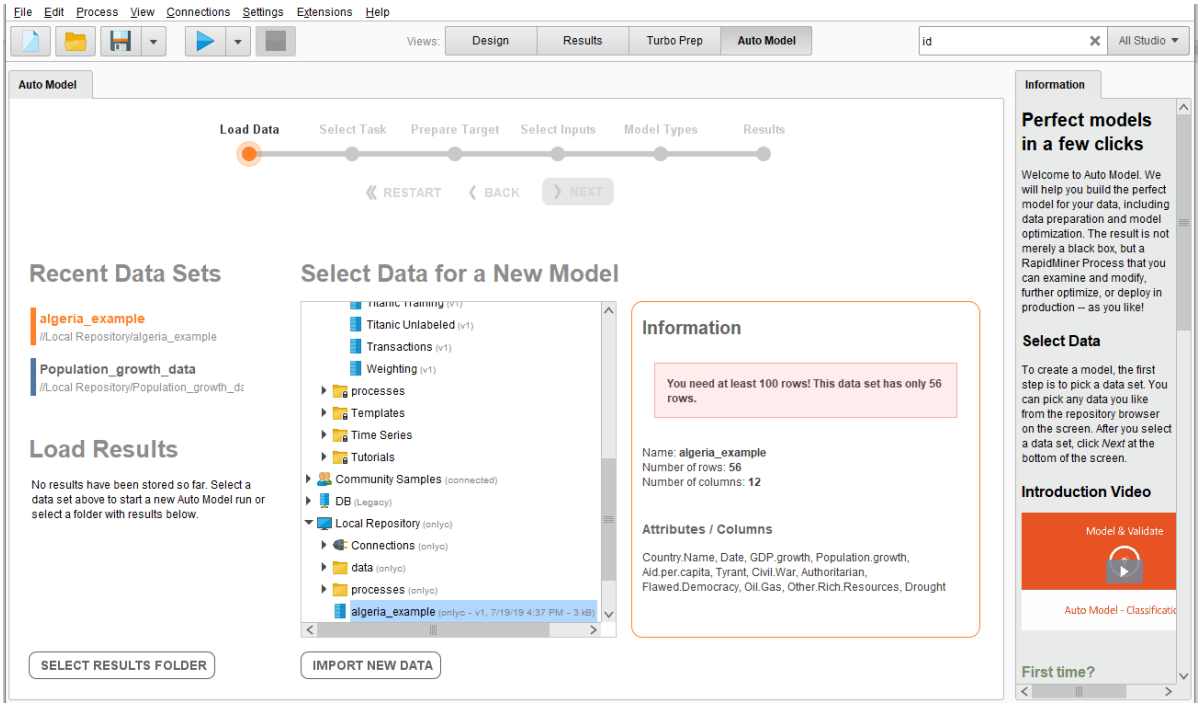
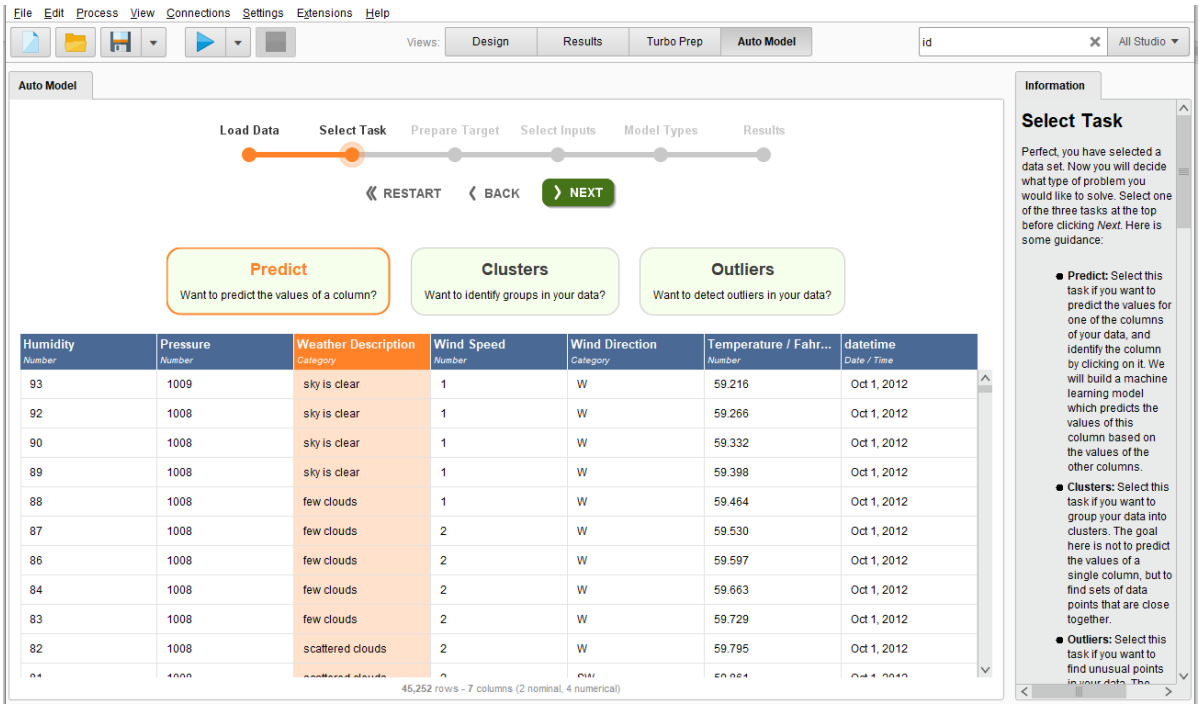


Figure 6-24 RapidMiner Auto Model extension

With at least 100 rows of data, the user can begin the analysis and is brought to the “Select Task” stage in Figure 6-25. There the user can choose between predicting, looking for clusters, and looking for outliers. I will only evaluate the “Predict” option.



Humidity Number	Pressure Number	Weather Description Category	Wind Speed Number	Wind Direction Category	Temperature / Fahr... Number	datetime Date / Time
93	1009	sky is clear	1	W	59.216	Oct 1, 2012
92	1008	sky is clear	1	W	59.266	Oct 1, 2012
90	1008	sky is clear	1	W	59.332	Oct 1, 2012
89	1008	sky is clear	1	W	59.398	Oct 1, 2012
88	1008	few clouds	1	W	59.464	Oct 1, 2012
87	1008	few clouds	2	W	59.530	Oct 1, 2012
86	1008	few clouds	2	W	59.597	Oct 1, 2012
84	1008	few clouds	2	W	59.663	Oct 1, 2012
83	1008	few clouds	2	W	59.729	Oct 1, 2012
82	1008	scattered clouds	2	W	59.795	Oct 1, 2012

Figure 6-25 RapidMiner Auto Model Select Task subsection

Next is prepare target, which shows an overview of the variable to be predicted with a graph. This graph adapts to the type of data, like in Figure 6-26, unlike in my application were the only available graph is a line graph that cannot show categorical data in a meaningful way. The user can also rename the classes of a categorical variable in this section.

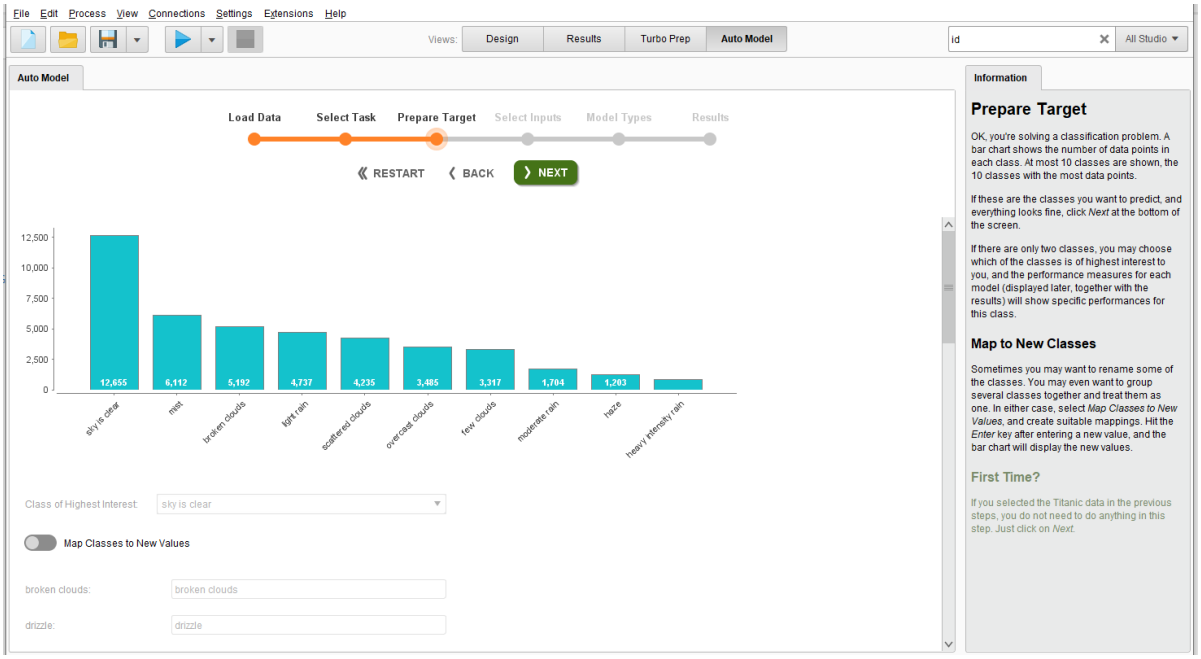


Figure 6-26 RapidMiner Auto Model Prepare Target subsection

The “Select Inputs” section is for quick automatic data preparation, as shown in Figure 6-27. If the user has already been trough Turbo Prep and used the cleansing feature, then this step in the Auto Model procedure is already completed.

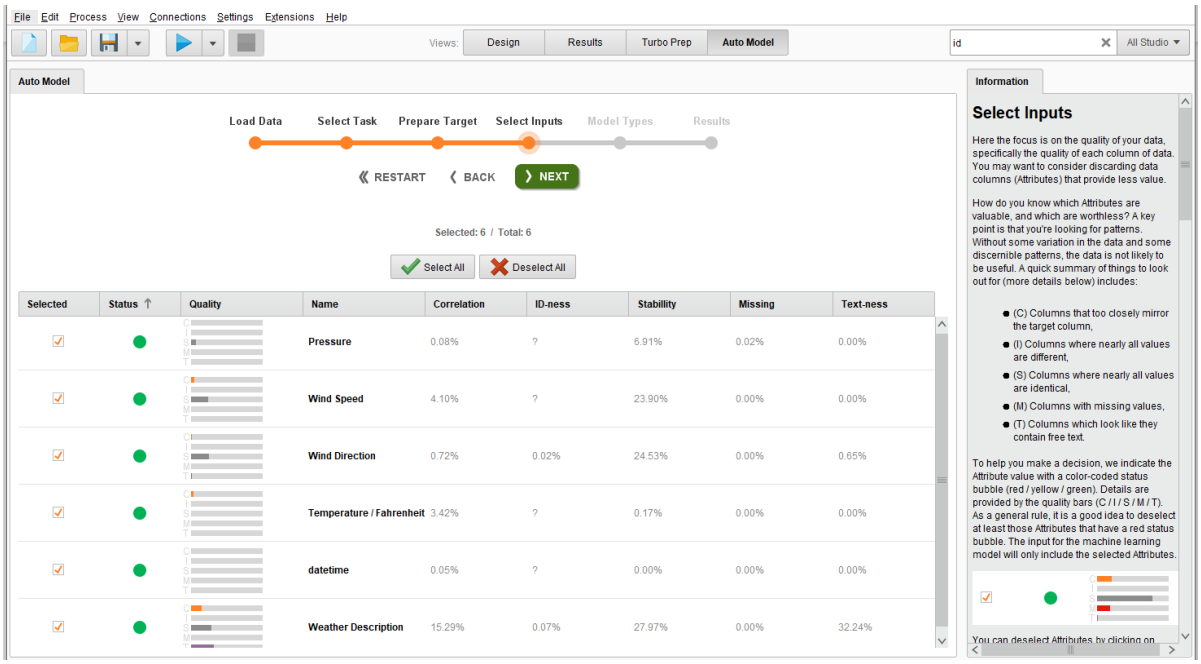


Figure 6-27 RapidMiner Auto Model Select Inputs subsection

The final step before initializing the automatic machine learning is the “Model Types” subsection, shown in Figure 6-28, where the user can select which models the algorithm can try and several more advanced options. A warning is displayed if a particular feature is particularly computationally heavy, such as the exceptionally useful automatic feature selection and automatic feature generation options, which should be enabled if the user desires maximum accuracy. The “Extract Date Information” option functions like the tk_augment_timeseries_signature function from the **timetk** package in my application (Dancho & Vaughan, 2018).

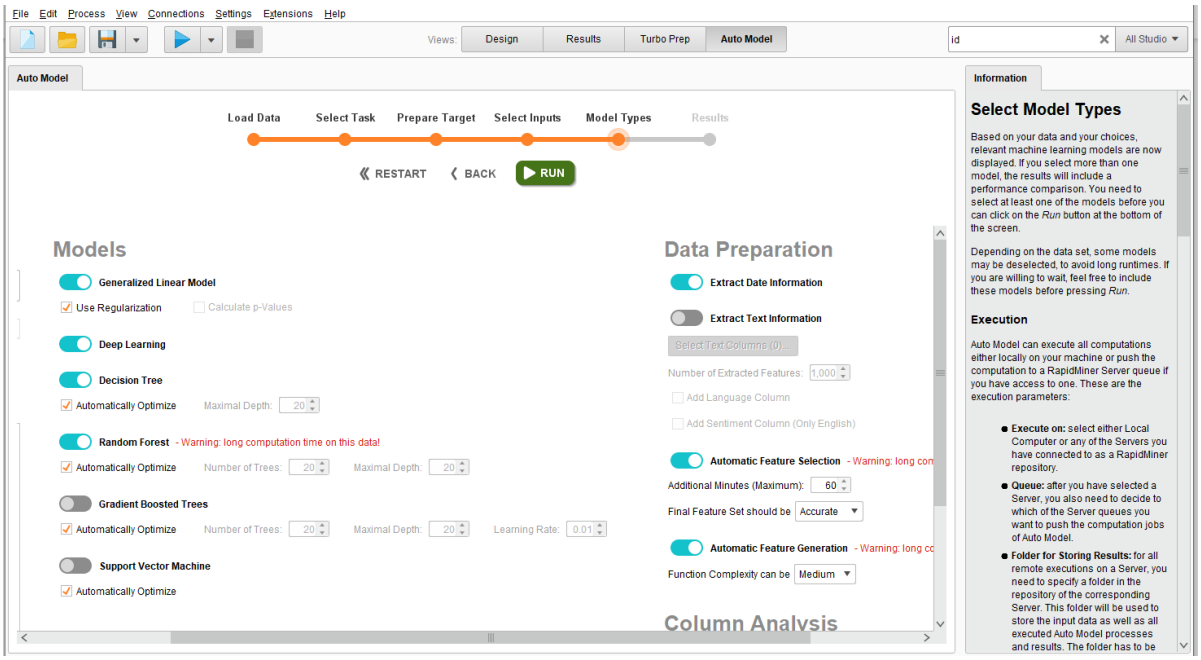


Figure 6-28 RapidMiner Auto Model Model Types subsection

When the user clicks “Run”, they are brought to the Results section where the progress bar in the top of the display works almost exactly like my progress notification, a spinning cake diagram lets the user know that the program is working, but other than that it only increments when it is finished with a type of prediction model. Since there have been no options for selecting how far into the future the user wants to predict, it creates a single prediction model.

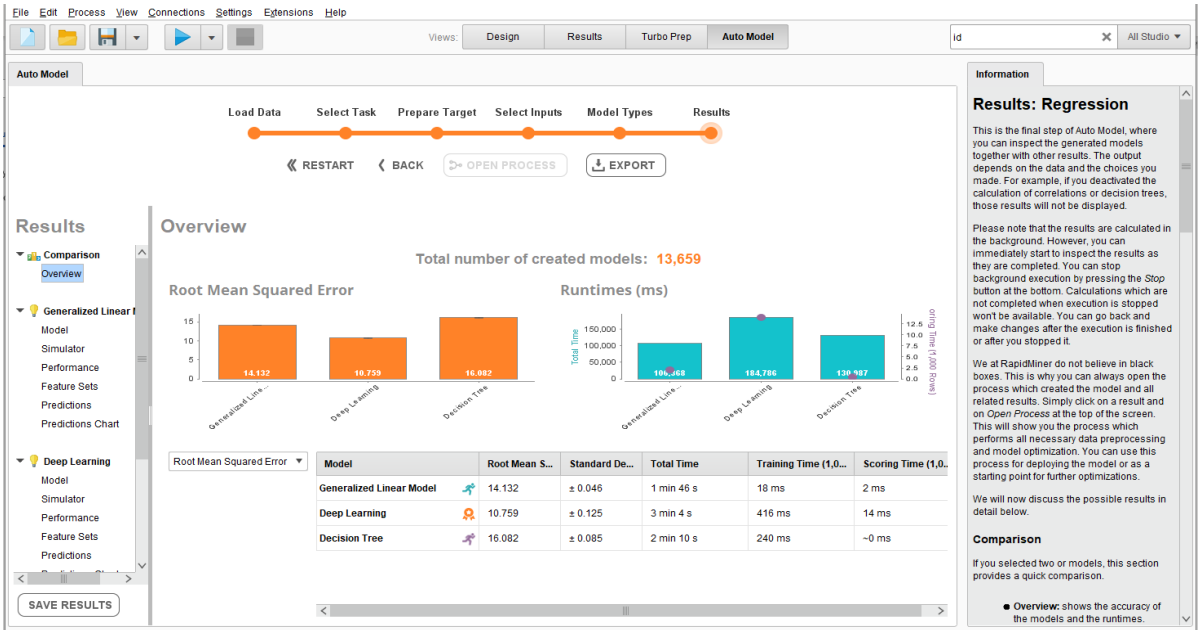


Figure 6-29 RapidMiner Auto Model Results subsection

An excellent feature of the Auto Model is that the user can evaluate completed models while the program is still working. When all models are completed, the key statistics are displayed, as in Figure 6-29, clearly showing which model is best. There are many different sub-menus for each of the model types that contain a wealth of information, much more than in the evaluation tab in my application. However, predictions are still limited to within the dataset, so unless the user has prepared the data with custom programming in the RapidMiner Studio Design system, it is impossible to extract predictions of the actual unobserved future.

The Rapid Miner Auto Model instead offers the simulator function, shown in Figure 6-30, in which users can experiment with changing the values of all the variables and seeing what each model predicts. A simulator would be handy in my application as well, but it fulfills a different purpose compared to plain predictions of future outcomes.

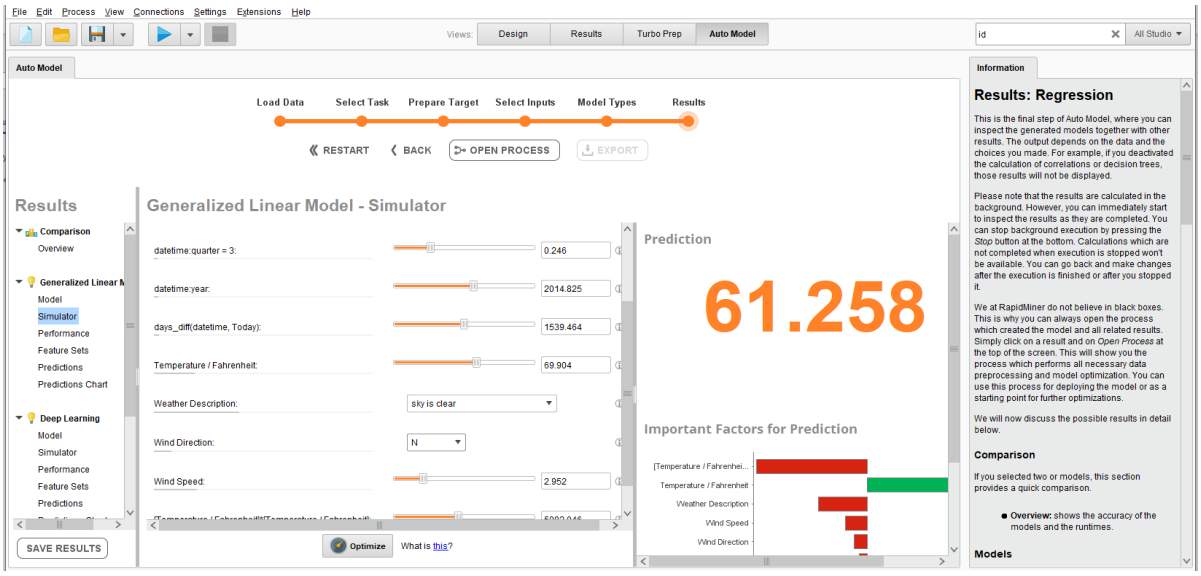


Figure 6-30 RapidMiner Auto Model Simulator function

6.4 Summary of comparison with RapidMiner

RapidMiner Studio with the Turbo Prep and Auto Model extensions is a powerful tool for data preparation and machine learning. However, the user must program the use of lags themselves to get predictions into the actual future. Other than the generation of lags and transposing of data, the upload and data preparation functionality in RapidMiner is vastly superior to my application. Further, the automatic machine learning is faster and much more transparent. Still, users who do not want to program cannot get the same time series prediction functionality out of RapidMiner.

7. Conclusion

For this thesis, I programmed an R based application for data preparation and the generation of predictions into the unobserved future, based on H2O.ai's open-source automatic machine learning algorithm. Using the shiny package, I made a web browser-based graphical user interface that enables users with no programming experience to utilize complicated R functions. While far from a finished product, the application delivers a lot of functionality, and the data preparation framework can be easily expanded with additional functions. Further, several improvements and additions should be made to the system for evaluating the prediction models, particularly the bootstrap calculation of confidence intervals for the predictions made by the best machine learning models, which was beyond the scope of what I managed to do in the time I had available. Support for panel data is also an important addition for future work.

The key concept behind how the actual predictions are generated is that lags back to the oldest period that is feasibly relevant for predicting the current value are generated for all variables. Then the machine learning models are only trained on data they would have available for prediction if each row was the newest observation. That means predictions can be generated without making any assumptions on the development of the independent variables or how the relationships differ over time. A key drawback is that this method is not very efficient because it multiplies the number of variables by the number of periods lagged and feeds everything to the AutoML algorithm, even though it is designed to handle large amounts of variables. Therefore, a method to "Cleanse" the data after the generation of lags, similar to the procedure in RapidMiner, would be beneficial. Still, the accuracy and quality of H2O AutoML has been documented thoroughly by other data scientists and is considered sufficient by the majority (Balaji & Allen, 2018), and my trick allows the algorithm to be used for time series predictions.

To evaluate my application, I sought to compare it to existing interfaces based on similar algorithms, but it seems like my solution to predictions into the unobserved future is rarer than I first assumed. When I found RapidMiner, I was certain that it would be easily able to fulfill the same functions as my application. It was only after investing a significant amount of time into their solution and looking at their related forums that I felt comfortable concluding that Turbo Prep and Auto Model could not perform all the specified tasks without assistance from manual programming. Making time series related predictions or forecasts now seem to be mostly in the purview of specialized machine learning algorithms, which are often designed for univariate time series. However, researching new types of algorithms and finding other

specialized products to compare with was also beyond the scope of this thesis for me, which is a clear disadvantage. There also seems to be a lack of automatic machine learning algorithms made for time series predictions that utilize different families of models in the way that H2O AutoML and RapidMiner Auto Model does.

The next natural step for my application would be working on the previously mentioned additions as well as finding companies that are interested in testing out such an analytical tool and using their feedback to improve. If the application ever gets to a stage where it could be commercialized, then I would rent a server with excellent computing capabilities, put my application there, and control the access with user registration and passwords, which is a supported feature in shiny.

References

Apache Spark. (2019). Home. Retrieved from https://spark.apache.org/

Arnold, J. B., Daroczi, G., Werth, B., Weitzner, B., Kunst, J., Auguie, B., Rudis, B., Wickham, H., Talbot, J., and London, J. (2019, May 13). ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'. Retrieved from https://CRAN.R-project.org/package=ggthemes

Chui, M., and Manyika J. (2015, March). "Competition at the digital edge: 'Hyperscale businesses,'" *McKinsey Quarterly*

Balaji, A. and Allen, A. (2018). "Benchmarking Automatic Machine Learning Frameworks." Retrieved from https://arxiv.org/pdf/1808.06492.pdf.

Bean, R. (2019, January 04). What We Learned From Top Execs About Their Big Data And AI Initiatives. Retrieved from https://www.forbes.com/sites/ciocentral/2019/01/02/what-we-learned-from-top-execs-about-their-big-data-and-ai-initiatives/

Brown, B., and Gottlieb J. (2016, April). "The need to lead in data and analytics," McKinsey & Company survey. Retrieved from http://www.mckinsey.com/business-functions/business-technology/our-insights/the-need-to-lead-in-dataand-analytics.

Chang, W., Cheng J., Allaire, J. J., Xie, Y, McPherson, J., RStudio, jQuery Foundation, jQuery contributors, jQuery UI contributors, Otto, M., Thornton J., Bootstrap contributors, Twitter, Inc., Farkas, A., Jehl, S., Petre, S., Rowls, A., Gandy, D., Reavis, B., Kowal, M. K., es5-shim contributors, Ineshin, D., Samhuri, S., SpryMedia Limited, Fraser J., Gruber, J., Sagalaev, I., and R Core Team. (2019, April 22). shiny: Web Application Framework for R. Available at https://CRAN.R-project.org/package=shiny

Dancho, M., and Vaughan, D. (2018, August 19). timetk: A Tool Kit for Working with Time Series in R. Available at https://CRAN.R-project.org/package=timetk

data. (2019). In *Merriam-Webster.com*. Retrieved April 28, 2019, from https://www.merriam- webster.com/dictionary/data

Displayr. (2019, June 02). Displayr/flipTime: Tools for Manipulating Dates and Time Series. Retrieved from https://rdrr.io/github/Displayr/flipTime/

Dobbs, R., Koller T., Ramaswamy S., Woetzel J., Manyika J., Krishnan R., and Andreula N. (2015, September) *Playing to win: The new global competition for corporate profits*, McKinsey Global Institute

DXC. (2019). Consulting. Retrieved from https://www.dxc.technology/consulting

Dryden Group. (2019, March 07). How Big Data is Changing the Way Companies Manage Indirect Expenses. Retrieved from https://drydengroup.com/big-data-changing-indirect-expenses/

Duhigg, C. (2012, February 16) How companies learn your secrets, The New York Times, Retrieved from https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html

FICO. (2019). Services. Retrieved from https://www.fico.com/en/platform/services

Fox, C. (2018, May 01). WORK INSTITUTE RELEASES NATIONAL EMPLOYEE RETENTION REPORT. Retrieved from https://workinstitute.com/about-us/news-events/articleid/2259/2018%20retention%20report

Frankenfield, J. (2019, April 18). Data Analytics. Retrieved April 28, 2019, from https://www.investopedia.com/terms/d/data-analytics.asp

Grace-Martin, K. (2019, March 20). Outliers: To Drop or Not to Drop. Retrieved from https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/

H2O.ai. (2019). Welcome to H2O 3. Retrieved from http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html

Henke, N., Bughin, J., Chui, M., Manyika, J., Saleh, T., Wiseman, B., and Sethupathy, G. (2016, December). The age of analytics: Competing in a data-driven world. Retrieved from https://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-age-of-analytics-competing-in-a-data-driven-world

Hester, J., Csárdi, G., Wickham, H., Chang, W., RStudio, Morgan, M., Tenenbaum, D., and Mango Solutions. (2019, June 24). remotes: R Package Installation from Remote Repositories, Including 'GitHub'. Retrieved from https://CRAN.R-project.org/package=remotes

Hitachi. (2019). Products. Retrieved from https://www.hitachivantara.com/en-us/products.html

Hyken, S. (2018, May 17). Businesses Lose $75 Billion Due To Poor Customer Service. Retrieved from https://www.forbes.com/sites/shephyken/2018/05/17/businesses-lose-75-billion-due-to-poor-customer-service/

IBM. (2019). Products. Retrieved from https://www.ibm.com/products

IDC Customer Spotlight (2011, March). Whirlpool corporation's digital detectives: Attensity provides the lens, retrieved from https://marketplace.informatica.com/mpresources/docs/Attensity_IDC_Spotlight.pdf

KNIME. (2019). Solutions. Retrieved from https://www.knime.com/solutions

Little, R. J., and Rubin, D. B. (2002). Statistical analysis with missing data. Wiley. *New York.*

LeDell, E. (2019). Kaggle competitions. Retrieved from https://www.kaggle.com/ledell

Microsoft. (2019). Services. Retrieved from https://www.microsoft.com/en-us/enterprise/services/design

Murphy, C. B. (2018, June 18). Why is social responsibility important to a business? Retrieved April 28, 2019, from https://www.investopedia.com/ask/answers/041015/why-social-responsibility-important-business.asp

Oracle. (2019). Consulting. Retrieved from https://www.oracle.com/consulting/

Picardo, E. (2019, February 02). Eight of the World's Top Companies Are American. Retrieved April 28, 2019, from https://www.investopedia.com/articles/active-trading/111115/why-all-worlds-top-10-companies-are-american.asp

PSL Corp. (2019). Home. Retrieved from https://www.pslcorp.com/

QBurst. (2019). Solutions. Retrieved from https://www.qburst.com/solutions/

Radware. (2019, January 15). Radware Report Shows That Respondents Claim Average Cost of Cyberattack Now Exceeds $1 Million. Retrieved from https://www.radware.com/newsevents/pressreleases/2019/average-cyberattack-exceeds-1-million

RapidMiner. (2019). Products. Retrieved from https://rapidminer.com/products/

RapidMiner. (2019). RapidMiner Studio Documentation. Retrieved from https://docs.rapidminer.com/latest/studio/

Rinker, T., Kurkiewicz, D., Hughitt, K., Wang, A., Aden-Buie, G., and Burk, L. (2019, March 11). pacman: Package Management Tool. Retrieved from https://CRAN.R-project.org/package=pacman

Rubin, D. B. (1987). Multiple imputation for survey nonresponse.

Sandeep. (2018, May 08). TOP 10 DATA ANALYTICS TOOLS. Retrieved from https://www.proschoolonline.com/blog/top-10-data-analytics-tools

SAS. (2019). All products. Retrieved from https://www.sas.com/en_us/software/all-products.html

Schaefer, M.W. (2012, March 07). Return On Influence: The Revolutionary Power of Klout, Social Scoring, and Influence Marketing, McGraw-Hill, pp 5-6.

Shu, C. (2014, January 26) Google acquires artificial intelligence startup DeepMind for more than $500 million, *TechCrunch.* Retrieved from https://techcrunch.com/2014/01/26/google-deepmind

Sinclair, T., M. (2015, September) *Beyond the talent shortage: How tech candidates search for jobs,* Indeed.com. Retrieved from http://offers.indeed.com/rs/699-SXJ-715/images/US-Beyond-the-Talent-Shortage.pdf

Sisense. (2019). Analysis Platforms. Retrieved from https://www.sisense.com/product/analyze/

Spinu, V., Grolemund, G., Wickham, H., Lyttle, I., Constigan, I., Law, J., Mitarotonda, D., Larmarange, J., Boiser, J., and Lee, C. H. (2018, April 11). lubridate: Make Dealing with Dates a Little Easier. Retrieved from https://CRAN.R-project.org/package=lubridate

Swalin, A. (2018, January 31). How to Handle Missing Data. Retrieved from https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

SoftwareMill. (2019). Services. Retrieved from https://softwaremill.com/services/

Tableau. (2019). Solutions. Retrieved from https://www.tableau.com/solutions

Teradata. (2019). Products. Retrieved from https://www.teradata.com/Products

Tukey, J. W. (1962). The future of data analysis. *The Annals of mathematical statistics*, *33*(1), 1-67.

Vesset, D., Gopal, C., Olofson, C. W., Bond, S., Fleming, M., and Schubmehl, D. (2018, September). *Worldwide Big Data and Analytics Software Market Shares, 2017: Healthy Growth Across the Board*, IDC.

Volini, E., Schwartz, J., Roy, I., Hauptmann, M., Van Durme, Y., Denny, B., and Bersin, J. (2019, February 21). *Global Human Capital Trends— HR cloud*, Deloitte Insights.

Wickham, H. (2017, December 11). reshape2: Flexibly Reshape Data: A Reboot of the Reshape Package. Retrieved from https://CRAN.R-project.org/package=reshape2

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C. Woo, K., Yutani, H., and RStudio. (2019, June 16). ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics. Retrieved from https://CRAN.R-project.org/package=ggplot2

Wickham, H., François, R., Henry, J., Müller, K., and RStudio. (2019, May 14). dplyr: A Grammar of Data Manipulation. Retrieved from https://CRAN.R-project.org/package=dplyr

Wickham, H., Hester, J., Francois, R., R Core Team, RStudio, Jylänki, J., and Jørgensen M. (2018, December 21). readr: Read Rectangular Text Data. Retrieved from https://CRAN.R-project.org/package=readr

Wong, J. (2013, April 7). imputation. Retrieved from https://github.com/jeffwong/imputation

World Bank. (2019). Global data on GDP, population, climate change and foreign aid. Retrieved from https://data.worldbank.org/ [Accessed 18 Mar. 2019].

Yalif, G. (2018, July 04). How Chime Bank Is Using AI to Drive Growth and Open More Accounts. Retrieved from https://thefinancialbrand.com/73355/predictive-analytics-digital-banking-website-accounts/

Xie, Y., Cheng, J., Tan, X., Allaire, J. J., Girlich, M., Freedman, G., Ellis, Rauh, J., jQuery contributors, SpryMedia Limited, Reavis, B., Gersen, L., Szopka, B., and RStudio Inc. (2019, June 11). DT: A Wrapper of the JavaScript Library 'DataTables'. Retrieved from https://CRAN.R-project.org/package=DT

Appendix

The following appendix contains all the code for my application, the formatting is from R Markdown, straight into word, only without the extra effects.

```
'/
Interface
Anders Stykket Gran s144892
/'

rm(list=ls())

if("pacman" %in% rownames(installed.packages()) == FALSE) {install.packages("pacman")}
require(pacman)

pacman::p_load(reshape2, nlme, tidyverse, shiny, shinydashboard, DT, stringr, dplyr, readr,
               tidyquant, timetk, remotes, lubridate, data.table, RCurl, jsonlite, ggplot2, ggthemes)

if ("h2o" %in% rownames(installed.packages()) == FALSE) {
  install.packages("h2o", type="source", repos="http://h2o-release.s3.amazonaws.com/h2o/rel-yates/4/R")}

if("flipTime" %in% rownames(installed.packages()) == FALSE) {remotes::install_github("Displayr/flipTime")}
if("imputation" %in% rownames(installed.packages()) == FALSE) {remotes::install_github("jeffwong/imputation")}
require(flipTime)

require(imputation)

require(h2o)

#Programming the user interface as dashboard
ui <- dashboardPage(
  #Header with title
  dashboardHeader(title = "Assisted analysis"),

  #All code for the sidebar, just tab name and icons. Must macth with tabItem names
  dashboardSidebar(
    sidebarMenu(
      menuItem("Data upload", tabName = "dataUpload", icon = icon("box-open")),
      menuItem("Complete dataset", tabName = "CompleteDataset", icon = icon("th")),
      menuItem("Time series analysis", tabName = "Analysis", icon = icon("robot")),
      menuItem("Evaluation", tabName = "Evaluation", icon = icon("chart-line"))
    )), #End of dashboard sidebar code
```

```
  #All code for all dashboard tabs, divided into tab items
  dashboardBody(
    tabItems(
      #---------- Data upload tab --------------------
      # First tab is for uploading the intitial dataset and making basic m
odification that allow it to be combined with more datasets
      tabItem(tabName = "dataUpload",

              sidebarLayout(
                sidebarPanel(
                  #Selecting the delimiter for the next file upload
                  selectInput(inputId = "Base.delimiter.1", label = "Choos
e the delimiter of your dataset",
                              c("Colon :","Comma ,","Semicolon ;","Tab \t"
,"Custom, type in the exact delimiter in the text input below"),
                              selected = NULL, multiple = FALSE),

                  #Input for wether the file has headers in the first row
                  selectInput(inputId = "Base.header.1", label = "Does the
dataset have headers in the first row?",
                              c("TRUE","FALSE"), multiple = FALSE),

                  #Button for easy upload of file
                  fileInput("Base.dataset.1", "Load you initial dataset th
en press confirm"),

                  #Displying informative text to the user
                  "The confirm button both confirms the uploading of the d
ata and any of the edits below,
                  an edit can not be undone after it is confirmed",

                  br(),


                  actionButton(inputId = "Confirm.button.1", label = "Conf
irm"),

                  #Adding some blank space so the text won't crowd the con
firm button
                  br(),
                  br(),

                  #Displying informative text to the user
                  "The remaining functions need to be activated with the f
orward button,
                  use the backward button to undo all changes implemented
in the last step",



                  #Choosing how to modify variable
                  selectInput(inputId ="Mod.choice.rename.delete.1",
                              "Select a function to apply to the data. It
```

```
may need you to select variables from the field below
                                and/or write in your own input",
                                c("Add a new variable to the dataset, you ca
n set a value for all observations by writing it",
                                  "Collapse rows to mean, choose all identif
ying variables",
                                  "Collapse rows to first observation, choos
e all identifying variables",
                                  "Collapse rows to last observation, choose
all identifying variables",
                                  "Collapse rows to maximum observation, cho
ose all identifying variables",
                                  "Collapse rows to minimum observation, cho
ose all identifying variables",
                                  "(Advanced) Collapse rows with custom func
tion, choose all identifying variables and write
                                  the exact name of the function",
                                  "Delete this dataset to make room for a ne
w upload",
                                  "Expand \"Date\" variable to year, semeste
r, quarter, month, week and day columns",
                                  "Move selected variables to the beginning
of the data table",
                                  "Only keep rows that match a single exact
value for selected variables",
                                  "Remove all rows that contain a single exa
ct value for selected variables",
                                  "Remove rows with values greater than limi
t for selected variables",
                                  "Remove rows with values less than limit f
or selected variables",
                                  "Rename selected variable(s)", "Remove sel
ected variable(s) entierly",
                                  "Select the date variable, which will be r
enamed \"Date\"",
                                  "Transform data from wide to long form, se
lect possible ID variables from list",
                                  "\a",
                                  "\a"
                                ),
                                selected = "\a", multiple = FALSE),




                    #Creating a list of all variables in the current dataset
                    selectInput(inputId ="Variable.names.1", "Select variabl
e(s) in the dataset",c(0), selected = NULL, multiple = TRUE),


                    #Text input for renaming variables or deleting rows
                    textInput(inputId ="Rename.delete.input.1", label = "Wri
te in custom input, such as a number or word. Multiple seperate
```

```
                            elements need to be seperated by a single comm
a followed by a single space \", \""),


                  #Shows the run number/dataset that is being dispkayed
                  textOutput("Run.tracker.1"),

                  #Buttons meant to control which dataset to display and i
nitiate modifications
                  actionButton(inputId = "Backward.number.1", label = "Bac
kward"),
                  actionButton(inputId = "Forward.number.1", label = "Forw
ard"),

                  br(),
                  br(),

                  "The export button sends the data to the complete datase
t tab, where it can be modified futher and expanded.
                  To combine multiple datasets you must export, use the de
lete function to delete the data in the data upload tab,
                  upload a new dataset and combine them with the functions
in the complete dataset tab",

                  br(),

                  actionButton(inputId = "Export.button.1", label = "Expor
t")

                ), #End of sidebar panel

                mainPanel(
                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Ba
se.dataset.table.1"))
                )
      )), #End of first tab


      #---------- Complete dataset tab --------------------
      # Second tab content
      tabItem(tabName = "CompleteDataset",

              sidebarLayout(
                sidebarPanel(

                  #Displying informative text to the user
                  "The confirm button confirms the loading of data from th
e data upload tab and any of the edits below,
                  an edit can not be undone after it is confirmed",

                  br(),


                  actionButton(inputId = "Confirm.button.2", label = "Conf
```

```
irm"),

                    #Adding some blank space so the text won't crowd the con
firm button
                    br(),
                    br(),

                    #Displying informative text to the user
                    "The remaining functions need to be activated with the f
orward button,
                    use the backward button to undo all changes implemented
in the last step",



                    #Choosing how to modify variable
                    selectInput(inputId ="Mod.choice.rename.delete.2",
                                "Select a function to apply to the data. It
may need you to select variables from
                                the field below and/or write in your own inp
ut",
                                c("-----Complete dataset functions-----",
                                  "Add the rows of a new dataset to this dat
aset, both datasets must have the same number of variables
                                  and those variables must have identical na
mes. Use other edits first if that is not the case",
                                  "Add the variables in a new dataset to thi
s dataset, please select the identifying variables,
                                  such as name and date, and those variables
must be identically named in both datasets",
                                  "Fill in all missing numeric values in the
dataset using KNN imputation.
                                  This function might not work if you don't
have enough complete data",
                                  "Replace all of an exact value with NA's f
or the chosen variables",
                                  "Replace all values greater than or equal
to limit with NA's for the chosen variables",
                                  "Replace all values less than or equal to
limit with NA's for the chosen variables",
                                  "Select all variables that should be const
ant over time in your data, such as names",
                                  "Select all time related variables except
the \"Date\" variable",
                                  "Select all categorical variables",
                                  "Select the variable you want to predict",
                                  "Select the time period that best describe
s your data. Choose from separate menu below",
                                  "Write in the number of periods gap you wa
nt between the last observation and the first useful prediction",
                                  "Write in the number of periods into the f
uture you want to predict",
                                  "Write in the maximum number of past perio
ds that could be relevant for predicting your variable",
```

```
                                "\a",
                                "\a",
                                "-----Data upload functions-----",
                                "Add a new variable to the dataset, you ca
n set a value for all observations by writing it",
                                "Collapse rows to mean, choose all identif
ying variables",
                                "Collapse rows to first observation, choos
e all identifying variables",
                                "Collapse rows to last observation, choose
all identifying variables",
                                "Collapse rows to maximum observation, cho
ose all identifying variables",
                                "Collapse rows to minimum observation, cho
ose all identifying variables",
                                "(Advanced) Collapse rows with custom func
tion, choose all identifying variables and write
                                the exact name of the function",
                                "Delete this dataset to make room for a ne
w upload",
                                "Expand \"Date\" variable to year, semeste
r, quarter, month, week and day columns",
                                "Move selected variables to the beginning
of the data table",
                                "Only keep rows that match a single exact
value for selected variables",
                                "Remove all rows that contain a single exa
ct value for selected variables",
                                "Remove rows with values greater than limi
t for selected variables",
                                "Remove rows with values less than limit f
or selected variables",
                                "Rename selected variable(s)", "Remove sel
ected variable(s) entierly",
                                "Select the date variable, which will be r
enamed \"Date\"",
                                "Transform data from wide to long form, se
lect possible ID variables from list"
                              ),
                              selected = "\a", multiple = FALSE),



                  #Creating a list of all variables in the current dataset
                  selectInput(inputId ="Variable.names.2", "Select variabl
e(s) in the dataset",c(0), selected = NULL, multiple = TRUE),


                  #Text input for renaming variables or deleting rows
                  textInput(inputId ="Rename.delete.input.2", label = "Wri
te in custom input, such as a number or word. Multiple
                            seperate elements need to be seperated by a si
```

```
ngle comma followed by a single space \", \""),


                  selectInput(inputId ="Period.definition.2",
                              "Select what defines your date variable and
the relevant time period for your dataset.
                              This is only used along with the appropriate
function from above",
                              c("Years", "Semesters", "Quarters", "Months"
, "Weeks", "Days"),
                              multiple = FALSE),

                  #Shows the run number/dataset that is being dispkayed
                  textOutput("Run.tracker.2"),

                  #Buttons meant to control which dataset to display and i
nitiate modifications
                  actionButton(inputId = "Backward.number.2", label = "Bac
kward"),
                  actionButton(inputId = "Forward.number.2", label = "Forw
ard"),

                  br(),
                  br(),

                  actionButton(inputId = "Finalize.button.2", label = "Fin
alize"),
                  downloadButton(outputId = "Download2",label = "Download
as .csv"),

                  br(),
                  br(),

                  "Dowload a random sample of your data for inspection, wr
ite in the number of rows you want in the sample
                  before clicking the button",

                  br(),

                  downloadButton(outputId = "Download3",label = "Download
random sample as .csv")



                ), #End of sidebar panel

                mainPanel(
                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Ba
se.dataset.table.2")), #Displays the dataset
                  br(),
                  "Below are a series of conditions that must be met befor
e you can finalize the dataset for predictions into the future.
                  Follow the instructions by using the corresponding funct
ions until it says that the data is ready, the order is not
```

```
                  important. You can make a new selection if you make a mi
stake or leave the selection blank if that is accurate or you
                  wish to use the default for the options that have a defa
ult.",
                  br(),
                  br(),
                  textOutput(outputId = "Constant.variables"), #The consta
nt variables condition
                  br(),
                  textOutput(outputId = "Other.time.variables"), #The othe
r time variables condition
                  br(),
                  textOutput(outputId = "Categorical.variables"), #The cat
egorical variables condition
                  br(),
                  textOutput(outputId = "Prediction.variable"), #The predi
ction variable condition
                  br(),
                  textOutput(outputId = "Period.def"), #The period definit
ion condition
                  br(),
                  textOutput(outputId = "Gap"), #The gap condition
                  br(),
                  textOutput(outputId = "Prediction.periods"), #The predic
tion periods condition
                  br(),
                  textOutput(outputId = "Lags"), #The lags condition
                  br(),
                  textOutput(outputId = "Ready.condition") #The total read
y condition



                )
      )), #End of second tab

      #---------- Time series analysis tab --------------------

      tabItem(tabName = "Analysis",

              sidebarLayout(
                sidebarPanel(


                  #Displying informative text to the user
                  "The confirm button confirms the loading of data from th
e previous tab and updating the graph and table with predictions
                  once those have been computed",

                  br(),

                  actionButton(inputId = "Confirm.button.3", label = "Conf
irm"),
```

```
                  #Adding some blank space so the text won't crowd the con
firm button
                  br(),
                  br(),

                  selectInput(inputId ="Model.run.time",
                              "Select how long you are willing to let the
machine learning work for each prediction period.
                              The time spent directly influences the quali
ty of the models and larger datasets need more time to give good quality r
esults.
                              The default is one hour",
                              c("5 minutes", "10 minutes", "15 minutes", "
30 minutes", "1 hour", "2 hours", "3 hours", "4 hours"),
                              multiple = FALSE, selected = "1 hour"),

                  br(),

                  "The machine learning will run for a substantial amount
of time, as set in the menu above. Once you click the
                  button below a progress notification will appear in the
bottom right corner of the interface. Once the
                  computation is done you may press the confirm button to
see the actual predictions. After this confirmation
                  you may also go to the evaluation tab to evaluate the pe
rformace and accuracy of the predictions",
                  br(),

                  actionButton(inputId = "Start.button.3", label = "Begin
machine learning and model prediction"),

                  br(),
                  br(),

                  "This download button downloads the entire dataset with
all predictions. Write in the desired name and
                  finish with .csv in the prompt",

                  br(),

                  downloadButton(outputId = "Download4",label = "Download
as .csv")



                ), #End of sidebar panel

                mainPanel(

                  plotOutput("Prediction.variable.plot.3", click = "Plot.c
lick.3"),
                  verbatimTextOutput("Plot.click.info.3"),

                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Ba
```

```
se.dataset.table.3"))
                  )
                )), #End of third tab

      #---------- Model evaluation tab --------------------

      tabItem(tabName = "Evaluation",

              sidebarLayout(
                sidebarPanel(

                  #Creating a list of all generated prediction models for
evaluation
                  selectInput(inputId ="Variable.names.4", "Select the mod
el you want to evaluate",c(0), selected = NULL, multiple = FALSE),

                  br(),

                  "The table below shows the most important variables for
generating predictions from your selected model.
                  The percentage shows how much of each prediction that is
based on that variable. lag_0xx refers to how
                  many periods behind each variable is to their prediction
",

                  br(),


                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Im
portant.variables.table")),

                  br(),

                  "The table below shows all the different models that wer
e tested by the machine learning in order of performace,
                  with the top one being the one chosen to make prediction
s",

                  br(),

                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Mo
del.leaderboard.table"))

                ), #End of sidebar panel

                mainPanel(
                  plotOutput("Prediction.variable.plot.4", click = "Plot.c
lick.4"),
                  verbatimTextOutput("Plot.click.info.4"),

                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Ev
aluation.metrics.table")),

                  div(style = 'overflow-x: scroll',DT::dataTableOutput("Ba
```

```
se.dataset.table.4"))
                  )
               )) #End of fourth tab

      ) #End of all tabs
      ) #End of dashboard code
) #End of UI code

#R code to create actual outputs, all objects must be saved as output$"exa
ct Id" and you need to use render functions to create outputs
server = function(input, output, session) {

  h2o.init(nthreads = -1)

  #Increaseing the maximum size of loaded files to 100 MB
  options(shiny.maxRequestSize=100*1024^2)



  #---------- Very simple observers to control which datasets are shown to
the user --------------------
  observeEvent({input$Forward.number.1}, {
    run.number.1 <- 1
  })

  observeEvent({input$Backward.number.1}, {
    run.number.1 <- 0
  })

  observeEvent({input$Forward.number.2}, {
    run.number.2 <- 1
  })

  observeEvent({input$Backward.number.2}, {
    run.number.2 <- 0
  })



  #Creating a run number to indicate which dataset to output
  run.num.1 <- reactive({ if(input$Forward.number.1-input$Backward.number.
1==0)
    0
    else isolate(run.number.1)
  })


  #Same for second tab
  run.num.2 <- reactive({ if(input$Forward.number.2-input$Backward.number.
2==0)
    0
    else isolate(run.number.2)
  })
```

```
  #Text to show the user which dataset is shown for 1st tab
  output$Run.tracker.1 <- renderText({ if(run.num.1()==0)
    "Showing saved dataset" else "Showing changed dataset"
  })

  #Text to show the user which dataset is shown for 2nd tab
  output$Run.tracker.2 <- renderText({ if(run.num.2()==0)
    "Showing saved dataset" else "Showing changed dataset"
  })


  #---------- Creating the basis for iterative changes ------------------
-


  #Creating reactiveValues object to hold my reactive datasets
  datasetInput.1 <- reactiveValues()
  datasetInput.2 <- reactiveValues()
  datasetInput.3 <- reactiveValues()
  datasetInput.4 <- reactiveValues()

  #Reads the dataset
  observeEvent(input$Base.dataset.1, {try({

    req(input$Base.dataset.1, cancelOutput = TRUE)

    datafile <- input$Base.dataset.1
    Dataset.base.1 <<- as_tibble(read_delim(datafile$datapath, col_names =
as.logical(input$Base.header.1), delim = {
      if(input$Base.delimiter.1=="Custom, type in the exact delimiter in t
he text input below")
      {req(input$Rename.delete.input.1, cancelOutput = TRUE)
        input$Rename.delete.input.1}
      else str_sub(input$Base.delimiter.1,-1,-1)
    },
    na = c("","NA","N/A","N\\A","NR")))

  })})



  #Use a seperate eventReactive varaible to allowfor iterative changes wit
hout infinite recursion
  datasetInput.1$base <- eventReactive(input$Confirm.button.1, try(Dataset
.base.1))
  datasetInput.2$base <- eventReactive(input$Confirm.button.2, try(Dataset
.base.2))
  datasetInput.3$base <- eventReactive(input$Confirm.button.3, try(Dataset
.base.3))
  datasetInput.4$base <- eventReactive(input$Confirm.button.3, try(Dataset
.base.3))
```

```
  #An observer saves the changes to the dataset with the confirm button
  observeEvent(input$Confirm.button.1, { try({

    if(run.num.1()==1)

    {req(datasetInput.1$change(), cancelOutput = TRUE)

      Dataset.base.1 <<- datasetInput.1$change()}

  })})


  #Same for tab 2
  observeEvent(input$Confirm.button.2, {try({

    if (run.num.2()==1)

    {req(Dataset.base.2, cancelOutput = TRUE)

      req(datasetInput.2$change(), cancelOutput = TRUE)

      Dataset.base.2 <<- datasetInput.2$change()}
  })})



  #The export tab creates the dataset for the second tab, though it must s
till be confirmed in the 2nd tab.
  observeEvent(input$Export.button.1, {try({

    req(Dataset.base.1, cancelOutput = TRUE)

    req(datasetInput.1$base(), cancelOutput = TRUE)

    Dataset.base.2 <<- datasetInput.1$base()


  })})

  #Finalize sends the completed dataset over to the time series analysis
  observeEvent(input$Finalize.button.2, {try({

    req(Dataset.base.2, cancelOutput = TRUE)

    req(datasetInput.2$base(), cancelOutput = TRUE)

    req({!is.null(Prediction.variable()) & !is.null(Period.def())} #The sa
me contitions as for the user prompt

        , cancelOutput = TRUE)

    Gap.actual <<- if (any(is.null(User.gap),is.na(User.gap))) {0} else {a
s.integer(User.gap)} #Default is 0 for gap.
```

```
    Prediction.periods.actual <<- if(any(is.null(User.prediction.periods),
is.na(User.prediction.periods))) {1} else
    {as.integer(User.prediction.periods)} #Default is 1 for prediction per
iods

    #Default is 10, but there must at least be as many lags as the gap plu
s prediction periods
    Lags.amounts <<- if (any(is.null(User.lags),is.na(User.lags))) {max((G
ap.actual+Prediction.periods.actual),10)}
    else {max((Gap.actual+Prediction.periods.actual),User.lags)}

    Lags.actual <<- (1+Gap.actual):Lags.amounts

    #Generates values for the Date variable. This is important because it
also affects the time series signature,
    #which could be important for predictions
    Extended.time <<- {Dataset.base.2[[nrow(Dataset.base.2),"Date"]] +
    {if (User.period.def=="Years") {years(1:(Gap.actual+Prediction.periods
.actual))}
      else if ((User.period.def=="Semesters")) {months((1:(Gap.actual+Pred
iction.periods.actual))*6)}
      else if ((User.period.def=="Quarters")) {months((1:(Gap.actual+Predi
ction.periods.actual))*3)}
      else if ((User.period.def=="Months")) {months(1:(Gap.actual+Predicti
on.periods.actual))}
      else if ((User.period.def=="Weeks")) {days((1:(Gap.actual+Prediction
.periods.actual))*7)}
      else if ((User.period.def=="Days")) {days(1:(Gap.actual+Prediction.p
eriods.actual))}
      else req(FALSE, cancelOutput = TRUE)
    }}

    #The rows that have been added to the data in order to generate predic
tions
    Prediction.rows <<- (nrow(Dataset.base.2)+1):(nrow(Dataset.base.2)+Gap
.actual+Prediction.periods.actual)


    #Extending the dataset
    Dataset.base.3 <<- {

      working.dataset <- Dataset.base.2

      working.dataset[Prediction.rows,] <- NA #First just adding blank row
s for the extra rows

      #Constant variables have their first value repeated in the extended
set as well
      working.dataset[Prediction.rows, User.constant.variables] <- working
.dataset[1,User.constant.variables]

      working.dataset[Prediction.rows, "Date"] <- Extended.time # The exte
nded dates are those generated previously
```

```
      working.dataset}

  })}) #End of the observe event and try functions



  #Creating a function to make the tables with statistics for evaluating e
ach model

  #Evaluation dataset




  #---------- Change dataset for data upload tab --------------------


  datasetInput.1$change <- eventReactive(
    input$Forward.number.1, try({
    {if(run.num.1()==1) {
      working.dataset <- datasetInput.1$base()

      if(input$Mod.choice.rename.delete.1=="Select the date variable, whic
h will be renamed \"Date\""){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset[,input$Variable.names.1] <- AsDate(unlist(working.
dataset[,input$Variable.names.1]))

        names(working.dataset)[names(working.dataset) == input$Variable.na
mes.1] <- "Date"

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.1=="Expand \"Date\" variable
to year, semester, quarter, month, week and day columns"){


        working.dataset <- working.dataset %>%
          dplyr::mutate(Year = lubridate::year(Date),
                        Semester = lubridate::semester(Date),
                        Quarter = lubridate::quarter(Date),
                        Month = lubridate::month(Date),
                        Week = lubridate::week(Date),
                        Day = lubridate::day(Date))


        working.dataset
      }
```

```
      else if(input$Mod.choice.rename.delete.1=="Collapse rows to mean, ch
oose all identifying variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% group_by_at(input$Variable.
names.1) %>% summarise_each(funs(mean))

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.1=="Collapse rows to first ob
servation, choose all identifying variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% group_by_at(input$Variable.
names.1) %>% summarise_each(funs(first))

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.1=="Collapse rows to last obs
ervation, choose all identifying variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% group_by_at(input$Variable.
names.1) %>% summarise_each(funs(last))

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.1=="Collapse rows to maximum
observation, choose all identifying variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% group_by_at(input$Variable.
names.1) %>% summarise_each(funs(max))

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.1=="Collapse rows to minimum
observation, choose all identifying variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% group_by_at(input$Variable.
names.1) %>% summarise_each(funs(min))
```

```
        working.dataset
      }

      else if(input$Mod.choice.rename.delete.1=="(Advanced) Collapse rows
with custom function, choose all identifying variables and write
              the exact name of the function"){

        req(input$Variable.names.1, cancelOutput = T)

        req(input$Rename.delete.input.1, cancelOutput = T)

        working.dataset <- working.dataset %>% group_by_at(input$Variable.
names.1) %>% summarise_each(get(input$Rename.delete.input.1))

        working.dataset
    }

      else if(input$Mod.choice.rename.delete.1=="Move selected variables t
o the beginning of the data table"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset[, c(input$Variable.names.1, set
diff(names(working.dataset), input$Variable.names.1))]

        working.dataset
      }



      else if(input$Mod.choice.rename.delete.1=="Rename selected variable(
s)"){

        req(input$Variable.names.1, cancelOutput = T)

        variable.matches <- grepl(paste(input$Variable.names.1 ,collapse
= "|"),variable.names(working.dataset))

        colnames(working.dataset)[variable.matches] <- unlist(strsplit(inp
ut$Rename.delete.input.1, ", "))

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.1=="Remove selected variable(
s) entierly"){

        req(input$Variable.names.1, cancelOutput = T)

        variable.matches <- grepl(paste(input$Variable.names.1 ,collapse
= "|"),variable.names(working.dataset))
        working.dataset <- working.dataset[,!variable.matches]
```

```
        working.dataset

      }

      else if(input$Mod.choice.rename.delete.1=="Remove rows with values g
reater than limit for selected variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% filter_at(vars(one_of(input
$Variable.names.1)),
                                                          all_vars(. <= as.
integer(input$Rename.delete.input.1)))

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.1=="Remove rows with values l
ess than limit for selected variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% filter_at(vars(one_of(input
$Variable.names.1)),
                                                          all_vars(. >= as.
integer(input$Rename.delete.input.1)))

        working.dataset

      }

      else if(input$Mod.choice.rename.delete.1=="Only keep rows that match
a single exact value for selected variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% filter_at(vars(one_of(input
$Variable.names.1)),
                                                          all_vars(. == inp
ut$Rename.delete.input.1))
        working.dataset

      }

      else if(input$Mod.choice.rename.delete.1=="Remove all rows that cont
ain a single exact value for selected variables"){

        req(input$Variable.names.1, cancelOutput = T)

        working.dataset <- working.dataset %>% filter_at(vars(one_of(input
$Variable.names.1)),
                                                          all_vars(. != inp
ut$Rename.delete.input.1))
        working.dataset
```

```
      }

      else if(input$Mod.choice.rename.delete.1=="Transform data from wide
to long form, select possible ID variables from list"){


        working.dataset <- melt(working.dataset, id.vars = input$Variable.
names.1)

        working.dataset

      }

      else if(input$Mod.choice.rename.delete.1=="Add a new variable to the
dataset, you can set a value for all observations by writing it"){

        req(input$Rename.delete.input.1, cancelOutput = T)

        new.var <- input$Rename.delete.input.1
        working.dataset <- cbind(working.dataset, new.var)

        working.dataset

      }


      else if(input$Mod.choice.rename.delete.1==  "Delete this dataset to
make room for a new upload"){

        working.dataset <- NULL

        working.dataset

      }

      else working.dataset #Final option if none of the alternatives are s
elected

    } else req(FALSE, cancelOutput = TRUE) #Prevents the data from changin
g in case the user manages to activate the function outside my setup
    }}))



  #Create current dataset

  current.Dataset.1 <- reactive(if(run.num.1()==0) {datasetInput.1$base()}
                                else {datasetInput.1$change()})


  #Same for advanced dataset
  output$Base.dataset.table.1 = DT::renderDataTable(current.Dataset.1())
```

```
  #---------- Change dataset for complete dataset tab -------------------

  datasetInput.2$change <- eventReactive(
    input$Forward.number.2, try({
    {if(run.num.2()==1) {
      working.dataset <- datasetInput.2$base()


      if(input$Mod.choice.rename.delete.2==  "Add the rows of a new datase
t to this dataset, both datasets must have the same number of variables
         and those variables must have identical names. Use other edits fi
rst if that is not the case"){

        req({paste(variable.names(Dataset.base.2), collapse = ",")==paste(
variable.names(Dataset.base.1),collapse = ",")}, cancelOutput = T)

        try(working.dataset <- rbind(Dataset.base.2, Dataset.base.1))

        working.dataset

    }

      else if(input$Mod.choice.rename.delete.2== "Add the variables in a n
ew dataset to this dataset, please select the identifying variables,
              such as name and date, and those variables must be identical
ly named in both datasets"){

        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- merge(Dataset.base.2, Dataset.base.1, by =
input$Variable.names.2, all.x = TRUE))

        working.dataset

    }


      else if(input$Mod.choice.rename.delete.2==  "Fill in all missing num
eric values in the dataset using KNN imputation.
              This function might not work if you don't have enough comple
te data"){

        only.numeric.data <- as.matrix(select_if(working.dataset, is.numer
ic))

        only.numeric.data <- kNNImpute(only.numeric.data, k = 10)

        only.numeric.data <- as_tibble(only.numeric.data$x)

        working.dataset[,names(only.numeric.data)] <- only.numeric.data

        working.dataset

    }
```

```
      else if(input$Mod.choice.rename.delete.2=="Select the date variable,
which will be renamed \"Date\""){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset[,input$Variable.names.2] <- AsDate(unlist(wor
king.dataset[,input$Variable.names.2]))

        names(working.dataset)[names(working.dataset) == input$Variable.na
mes.2] <- "Date"})

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.2=="Expand \"Date\" variable
to year, semester, quarter, month, week and day columns"){


        try(working.dataset <- working.dataset %>%
              dplyr::mutate(Year = lubridate::year(Date),
                            Semester = lubridate::semester(Date),
                            Quarter = lubridate::quarter(Date),
                            Month = lubridate::month(Date),
                            Week = lubridate::week(Date),
                            Day = lubridate::day(Date)))

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.2=="Collapse rows to mean, ch
oose all identifying variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- working.dataset %>% group_by_at(input$Varia
ble.names.2) %>% summarise_each(funs(mean)))

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.2=="Collapse rows to first ob
servation, choose all identifying variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- working.dataset %>% group_by_at(input$Varia
ble.names.2) %>% summarise_each(funs(first)))

        working.dataset
      }
```

```
      else if(input$Mod.choice.rename.delete.2=="Collapse rows to last obs
ervation, choose all identifying variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- working.dataset %>% group_by_at(input$Varia
ble.names.2) %>% summarise_each(funs(last)))

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.2=="Collapse rows to maximum
observation, choose all identifying variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- working.dataset %>% group_by_at(input$Varia
ble.names.2) %>% summarise_each(funs(max)))

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.2=="Collapse rows to minimum
observation, choose all identifying variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- working.dataset %>% group_by_at(input$Varia
ble.names.2) %>% summarise_each(funs(min)))

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.2=="(Advanced) Collapse rows
with custom function, choose all identifying variables and write
              the exact name of the function"){

        req(input$Variable.names.2, cancelOutput = T)

        req(input$Rename.delete.input.2, cancelOutput = T)

        try(working.dataset <- working.dataset %>% group_by_at(input$Varia
ble.names.2) %>% summarise_each(get(input$Rename.delete.input.2)))

        working.dataset
    }

      else if(input$Mod.choice.rename.delete.2=="Move selected variables t
o the beginning of the data table"){
```

```
        req(input$Variable.names.2, cancelOutput = T)

        try(working.dataset <- working.dataset[, c(input$Variable.names.2,
setdiff(names(working.dataset), input$Variable.names.2))])

        working.dataset
      }



      else if(input$Mod.choice.rename.delete.2=="Rename selected variable(
s)"){

        req(input$Variable.names.2, cancelOutput = T)

        try({variable.matches <- grepl(paste(input$Variable.names.2 ,colla
pse  = "|"),variable.names(working.dataset))

        colnames(working.dataset)[variable.matches] <- unlist(strsplit(inp
ut$Rename.delete.input.2, ", "))})

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.2=="Remove selected variable(
s) entierly"){

        req(input$Variable.names.2, cancelOutput = T)

        try({variable.matches <- grepl(paste(input$Variable.names.2 ,colla
pse  = "|"),variable.names(working.dataset))
        working.dataset <- working.dataset[,!variable.matches]})

        working.dataset

      }


      else if(input$Mod.choice.rename.delete.2=="Replace all of an exact v
alue with NA's for the chosen variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% mutate_at(vars(one_of(
input$Variable.names.2)),
                                                               list(~na_if(
., input$Rename.delete.input.2)))})
        working.dataset

      }


      else if(input$Mod.choice.rename.delete.2=="Replace all values greate
r than or equal to limit with NA's for the chosen variables"){
```

```
        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% mutate_at(vars(one_of(
input$Variable.names.2)),
                                                               list(~ ifels
e( . >= as.integer(input$Rename.delete.input.2), NA, .)))})

        working.dataset
      }


      else if(input$Mod.choice.rename.delete.2=="Replace all values less t
han or equal to limit with NA's for the chosen variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% mutate_at(vars(one_of(
input$Variable.names.2)),
                                                               list(~ ifels
e( . <= as.integer(input$Rename.delete.input.2), NA, .)))})

        working.dataset

      }



      else if(input$Mod.choice.rename.delete.2=="Remove rows with values g
reater than limit for selected variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% filter_at(vars(one_of(
input$Variable.names.2)),
                                                               all_vars(. <
= as.integer(input$Rename.delete.input.2)))})

        working.dataset
      }

      else if(input$Mod.choice.rename.delete.2=="Remove rows with values l
ess than limit for selected variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% filter_at(vars(one_of(
input$Variable.names.2)),
                                                               all_vars(. >
= as.integer(input$Rename.delete.input.2)))})

        working.dataset

      }
```

```
      else if(input$Mod.choice.rename.delete.2=="Only keep rows that match
a single exact value for selected variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% filter_at(vars(one_of(
input$Variable.names.2)),
                                                              all_vars(. =
= input$Rename.delete.input.2))})
        working.dataset

      }

      else if(input$Mod.choice.rename.delete.2=="Remove all rows that cont
ain a single exact value for selected variables"){

        req(input$Variable.names.2, cancelOutput = T)

        try({working.dataset <- working.dataset %>% filter_at(vars(one_of(
input$Variable.names.2)),
                                                              all_vars(. !
= input$Rename.delete.input.2))})
        working.dataset

      }

      else if(input$Mod.choice.rename.delete.2=="Transform data from wide
to long form, select possible ID variables from list"){


        try(working.dataset <- melt(working.dataset, id.vars = input$Varia
ble.names.2))

        working.dataset

      }

      else if(input$Mod.choice.rename.delete.2=="Add a new variable to the
dataset, you can set a value for all observations by writing it"){

        req(input$Rename.delete.input.2, cancelOutput = T)

        try({new.var <- input$Rename.delete.input.2
        working.dataset <- cbind(working.dataset, new.var)})

        working.dataset

      }


      else if(input$Mod.choice.rename.delete.2==  "Delete this dataset to
make room for a new upload"){
```

```
        try(working.dataset <- NULL)

        working.dataset

      }

      else working.dataset #Final option if none of the alternatives are s
elected

    } else req(FALSE, cancelOutput = TRUE) #Prevents the data from changin
g in case the user manages to activate the function outside my setup
    }))




  #Creating current dataset that will be loaded for the user

  current.Dataset.2 <- reactive(if(run.num.2()==0) {datasetInput.2$base()}
                                else {datasetInput.2$change()})


  #The output object rendered as a table
  output$Base.dataset.table.2 = DT::renderDataTable(current.Dataset.2())



  #---------- Time series preperations --------------------


  #To avoid trouble with important user inputs not existing before first i
nteraction this simple observer simply sets all of
  #them to NULL if they don't exists yet and therefore avoiding using lots
of !exists conditions later

  observeEvent(input$Base.delimiter.1, {try({ #By having it trigger on a s
electInput input it starts immedeatly on loading the app

    User.imputs <- c("User.categorical.variables","User.constant.variables
","User.gap","User.lags","User.other.time.variables",
                     "User.period.def","User.prediction.periods","User.pre
diction.variable")

    for (i in 1:length(User.imputs)){

      if(!exists(User.imputs[i])) {assign(User.imputs[i], NULL, pos = .Glo
balEnv)}

    }
    #Only want this to run once, this is an extra safety there as the if s
tatement with !exists should prevent re-running as well.
  })}, once = TRUE)
```

```
  #Creating the global list of constant variables, meaning those who shoul
d not be lagged for prediction, but still be included
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2=="Select all variables that shoul
d be constant over time in your data, such as names",
        cancelOutput = TRUE) #The observer should only act in this case

    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
    User.constant.variables <<- input$Variable.names.2
  }) #No action without exact match

  Constant.variables <- eventReactive(input$Forward.number.2,{ #Have to do
the extra step of creating a reactive object in
    #between to avoid the value erasing itself and recursion
    {User.constant.variables}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the constant variables list. There are 2 sta
tes. No variables selected and some varaibles selected.
  output$Constant.variables <- renderText({if (is.null(Constant.variables(
)))
  {"Please select all variables that should not change over time, such as
names or constant identifiers, if there are
    any such variables. If your data does not have any such variables then
this step is completed."}

    else {paste0("You have selected: ", paste(Constant.variables(), collap
se = ", "),"" as your constant variable(s).
                 Make a new selection to overwrite this.")}
    })



  #Creating the global list of extra time variables, these will not be lag
ged and removed before actal analysis to avoid
  #overlap with the augment timeseries function
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2=="Select all time related variabl
es except the \"Date\" variable", cancelOutput = TRUE)
    #The observer should only act in this case
    User.other.time.variables <<- input$Variable.names.2
    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
  }) #No action without exact match

  Other.time.variables <- eventReactive(input$Forward.number.2,{
    #Have to do the extra step of creating a reactive object in between to
avoid the value erasing itself and recursion
```

```
    {User.other.time.variables}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the extra time variables list. There are 2 s
tates. No variables selected and some varaibles selected.
  output$Other.time.variables <- renderText({if (is.null(Other.time.variab
les()))
  {"Please select all variables that represent time or time periods, other
than the \"Date\" variable, if there are
    any such variables. If your data does not have any such variables then
this step is completed."}

    else {paste0("You have selected: ", paste(Other.time.variables(), coll
apse = ", "),
                 " as extra time variable(s). Make a new selection to over
write this.")}
  })



  #Creating the global list of categorical variables. These need to be for
matted as fators to be treated correctly
  #by the machine learning package
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2=="Select all categorical variable
s", cancelOutput = TRUE)
    #The observer should only act in this case
    User.categorical.variables <<- input$Variable.names.2
    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
  }) #No action without exact match

  Categorical.variables <- eventReactive(input$Forward.number.2,{
    #Have to do the extra step of creating a reactive object in between to
avoid the value erasing itself and recursion
    {User.categorical.variables}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the categorical variables list. There are 2
states. No variables selected and some varaibles selected.
  output$Categorical.variables <- renderText({if (is.null(Categorical.vari
ables()))
  {"Please select all variables that represent categories, such as yes/no,
binary 0/1, gender, etc., if there are any
    such variables. If your data does not have any such variables then thi
s step is completed."}

    else {paste0("You have selected: ", paste(Categorical.variables(), col
lapse = ", "),
                 " as categorical variable(s). Make a new selection to ove
rwrite this.")}
```

```
  })


  #Creating the global object with the prediction variable. These need to
be formatted as fators to be treated correctly
  #by the machine learning package
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2=="Select the variable you want to
predict", cancelOutput = TRUE)
    #The observer should only act in this case
    User.prediction.variable <<- input$Variable.names.2
    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
  }) #No action without exact match

  Prediction.variable <- eventReactive(input$Forward.number.2,{ #Have to d
o the extra step of creating a reactive object
    #in between to avoid the value erasing itself and recursion
    {User.prediction.variable}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the prediction variables list. There are 3 s
tates. No variable selected, more than one
  #selected and exactly one varaible selected.
  output$Prediction.variable <- renderText({if (is.null(Prediction.variabl
e()))
  {"Please select the single varaible that you want to predict. You can no
t proceed unless you select only one variable"}

    else if (length(Prediction.variable())>1) {"You have selected more tha
n one variable,
      please select only one variable to predict by overwriting this selec
tion."}

    else {paste0("You have selected: ", paste(Prediction.variable(), colla
pse = ", "),
                 " as your prediction variable. Make a new selection to ov
erwrite this.")}
    })


  #Creating the global object with the period definition. It affects the f
uture date variables and therefore their time signatures
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2=="Select the time period that bes
t describes your data. Choose from separate menu below",
        cancelOutput = TRUE) #The observer should only act in this case
    User.period.def <<- input$Period.definition.2  #Sets the global variab
le, by not choosing any varaibles the user can make it
    #like no variables were ever chosen
  }) #No action without exact match
```

```
  Period.def <- eventReactive(input$Forward.number.2,{
    #Have to do the extra step of creating a reactive object in between to
avoid the value erasing itself and recursion
    {User.period.def}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the categorical variables list. There are 3
states. No variable selected, more than one
  #selected and exactly one varaible selected.
  output$Period.def <- renderText({if (is.null(Period.def()))
  {"Please select the time period that most closely defines your data. Use
the function selector and the separate dropdown menu below."}

    else {paste0("You have selected: ", Period.def()," as your time period
definition. Make a new selection to overwrite this.")}
  })




  #Creating the global object with the gap number This decides how many pe
riods there should be between the last observation
  #and the first meaningful prediction model
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2==
  "Write in the number of periods gap you want between the last observatio
n and the first useful prediction", cancelOutput = TRUE)
    #The observer should only act in this case
    User.gap <<- {if(input$Rename.delete.input.2=="") NULL else as.integer
(input$Rename.delete.input.2)}
    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
  }) #No action without exact match

  Gap <- eventReactive(input$Forward.number.2,{ #Have to do the extra step
of creating a reactive object in between to avoid
    #the value erasing itself and recursion
    {User.gap}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the gap object. There are 3 states. Integer
input, <0 input, non integer input and no input.
  output$Gap <- renderText({if (is.null(Gap()))
  {"Please write in the number of periods gap you want in between the last
observation and the first meaningfull prediction.
    This is usefull if predictions for one or more immediate periods don't
have value because they have already happened or
    will happen too soon. The default is 0, don't select anything, leave i
t blank or write in \"0\" if you don't want any gap."}
```

```
    else if (is.na(Gap())) {"You written in something that could not be tr
anslated to a single number,
      please write in a single number to overwrite this."}

    else if (Gap()<0) {"You have written in a negative gap, please overwri
te this by writing in a gap of at least 0."}

    else {paste0("You have chosen to use: ", Gap()," period(s) gap between
the last observation and first important prediction.
                 Make a new selection to overwrite this.")}
    })



  #Creating the global object with the number of prediction periods. This
decides how many models will be created and
  #therefore it significantly increases computation time.
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2=="Write in the number of periods
into the future you want to predict", cancelOutput = TRUE)
    #The observer should only act in this case
    User.prediction.periods <<- {if(input$Rename.delete.input.2=="") NULL
else as.integer(input$Rename.delete.input.2)}
    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
  }) #No action without exact match

  Prediction.periods <- eventReactive(input$Forward.number.2,{
    #Have to do the extra step of creating a reactive object in between to
avoid the value erasing itself and recursion
    {User.prediction.periods}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the prediction periods. There are 4 states.
Integer input, <1 input, non integer input and no input.
  output$Prediction.periods <- renderText({if (is.null(Prediction.periods(
)))
  {"Please write in the number of periods, after the gap, that you want to
create predictions for. A machine learning model will be
    created for each period so this measure will drastically increase comp
uation times. For example, if you have a gap of 2 periods
    and want predictions for 4 periods, the final prediction will be for 6
periods into the future. Minimum is 1 period,
    no analysis would be performed for 0 periods."}

    else if (is.na(Prediction.periods())) {"You written in something that
could not be translated to a single number, please write
      in a single number to overwrite this."}

    else if (Prediction.periods()<=0) {"You have chosen to make prediction
s for less than 1 period. That amounts to no analysis and
```

```
      therefore the minimum is 1, please write in a single number greater
than 0 to overwrite this."}

    else {paste0("You have chosen to make predictions for: ", Prediction.p
eriods()," period(s) after the gap. Make a new selection
                 to overwrite this.")}
    })



  #Creating the global object with the number of lags. This decides how ma
ny lag variables will be created and therefore it
  #increases computation time.
  observeEvent(input$Forward.number.2,{ #Trigger it with the forward butto
n so that it works like other changes
    req(input$Mod.choice.rename.delete.2==
  "Write in the maximum number of past periods that could be relevant for
predicting your variable", cancelOutput = TRUE)
    #The observer should only act in this case
    User.lags <<- {if(input$Rename.delete.input.2=="") NULL else as.intege
r(input$Rename.delete.input.2)}
    #Sets the global variable, by not choosing any varaibles the user can
make it like no variables were ever chosen
  }) #No action without exact match

  Lags <- eventReactive(input$Forward.number.2,{
    #Have to do the extra step of creating a reactive object in between to
avoid the value erasing itself and recursion
    {User.lags}
  }, ignoreNULL = FALSE)


  #Creating user feedback for the prediction periods. There are 4 states.
Integer input, <1 input, non integer input and no input.
  output$Lags <- renderText({if (is.null(Lags()))
  {"Please write in the maximum number of past periods that could hold rel
evant information for predicting your variable.
    The machine learning will only use those that are actually usefull so
it is better to make a higher estimate. However,
    it also increases computation time so you should not push it beyond wh
at is plausible either. It needs to be at least
    as high as the gap pluss the number of prediction periods and the numb
er should not be a significant fraction of your
    number of rows. The default is an uneducated guess of 10 periods."}

    else if (is.na(Lags())) {"You written in something that could not be t
ranslated to a single number, please write in a single
      number to overwrite this."}

    else if (Lags()<=0) {"You have chosen to use less than 1 period histor
ical data for prediction. That amounts to no analysis
      and the minimum is still your gap pluss your prediction periods. Ple
ase write in a single number that satisfies the
```

```
      criteria to overwrite this."}

    else {paste0("You have chosen to use: ", Lags()," period(s) of histori
cal data to make predictions. Make a new selection
                 to overwrite this.")}
    })



  #Creating user feedback for wether the data is ready to be finalized for
time series analysis. The finalize button will not
  #function unless this condition is met. Only 2 states, either all condit
ions are met or at least one is not.
  output$Ready.condition <- renderText({if (!is.null(Prediction.variable()
) & !is.null(Period.def()))


  {"Your data is ready for time series predictions. The Finalize button wi
ll send the data to the Time Series Analysis tab."}

    else {"One or more conditions that don't have a default have not been
defined. Please complete the conditions as described
      above. The Finalize button will not do anything."}
    })



  #Making a simple reactive to help make sure the conditions actually inva
lidate and update with their new informations on
  #a single click of the forward button

  Condition.helper <- eventReactive(input$Forward.number.2,{

    Constant.variables()
    Other.time.variables()
    Categorical.variables()
    Prediction.variable()
    Period.def()
    Gap()
    Prediction.periods()
    Lags()
    NULL

  }, ignoreNULL = FALSE)



  #---------- Datasets for time series analysis tab --------------------



  #Creating current dataset that will be loaded for the user
```

```
  current.Dataset.3 <- reactive({
    working.dataset <- datasetInput.3$base() %>% select(one_of("Date",User
.constant.variables,"Predictions",User.prediction.variable))
    working.dataset[Prediction.rows,]
  })



  #The output object rendered as a table
  output$Base.dataset.table.3 = DT::renderDataTable(current.Dataset.3())

  #---------- Building the prediction models --------------------



  #This is the observer that actually starts the automl function

  observeEvent(input$Start.button.3, {try({

    working.dataset <- Dataset.base.3

    extended.dataset <- Dataset.base.3 #This non-global object is created
to interact with the functions contained in this observer

    extended.dataset <- add_column(extended.dataset, Predictions = NA,.bef
ore = User.prediction.variable)


    Non.lagg.variables <- c(User.constant.variables, "Date", User.other.ti
me.variables, "Predictions")
    #Lags have to be generated to create predictions, but lags of constant
variables and time varables are meaningless



    lag.names <- paste("lag", formatC(Lags.actual, width = 3, flag = "0"),
                       #Each lag is named after its base variable and then
the lag number with three integers, 007, 145 etc. That means
                       #the system does not support lags in excess of 999,
but that many lags does not seem reasonable in most cases
                       sep = "_")

    lag.functions <- setNames(paste("dplyr::lag(., ", Lags.actual, ")"), l
ag.names)
    #This function combines with the last one to give correct names

    lag.variables <- setdiff(names(working.dataset), Non.lagg.variables)
    #All variables that are not marked in the non-lag object will be lagge
d



    working.dataset <- working.dataset %>% mutate_at(vars(lag.variables),
funs_(lag.functions)) #This command implements the lags
```

```
    working.dataset <- select(working.dataset, setdiff(names(working.datas
et),User.other.time.variables))
    #Remove all other time variables to avoid overlap with augmented times
series signature


    #working.dataset <- tk_augment_timeseries_signature(working.dataset) #
Adds extra variables to help the machine learning
    #catch all time related effects


    #working.dataset <- working.dataset %>% mutate_if(is.character,as.fact
or) #Can't operate with character columns, set those to factors



    variable.lag.identifier <- as.integer(substr(names(working.dataset), n
char(names(working.dataset))-2, nchar(names(working.dataset))))
    #Extracts the lag number at the end of each name

    variable.lag.identifier[is.na(variable.lag.identifier)] <- 0 #The vari
ables without lag numbers are assumed to be lag 0


    evaluation.metrics <- tibble("Prediction model"=NA, "Apx. accuracy (R^
2)"=NA, "Mean Squared Error"=NA,
                                 "Root Mean Squared Error"=NA, "Mean Absol
ute Error"=NA)
    #Creating a non-global object to populate with the for loop before ass
igning the values to a global object
    evaluation.metrics[1:Prediction.periods.actual,] <-NA


    #Creating an object to hold the users time deinition.

    max.runtime <- {if(input$Model.run.time=="5 minutes") {300}
      else {if (input$Model.run.time=="10 minutes") {600}
        else {if (input$Model.run.time=="15 minutes") {900}
          else {if (input$Model.run.time=="30 minutes") {1800}
            else {if (input$Model.run.time=="1 hour") {3600}
              else {if (input$Model.run.time=="2 hours") {7200}
                else {if (input$Model.run.time=="3 hours") {10800}
                  else {14400}}}}}}}


    #Here comes the actual loop that will create a full h2o automl model f
or each prediction period, based on the users input.

    withProgress(message = 'Computing prediction models', value = 0, {

      for (i in (1+Gap.actual):(Gap.actual+Prediction.periods.actual)){ #T
he number of models is at least one and equal to the
```

```
        #sequence from 1+Gap to Gap+Prediction periods.

        incProgress(1/Prediction.periods.actual, detail = paste0("Working
on model with at least ", i, " periods old data"))


        assign(paste0("Dataset.model.",i),{ #The initial h2o object used t
o create the models and showing predictions on the actual data
          period.matches <- as.logical({variable.lag.identifier>=i}+{names
(working.dataset) %in% Non.lagg.variables}+
          {names(working.dataset) %in% User.prediction.variable})
          #Only the prediction variable, variables that should not be lagg
ed and lags of varaibles that would actually be
          #availiable for prediction for the models time horizon are inclu
ded

          temp <- working.dataset[complete.cases(working.dataset),period.m
atches]
          #Using simple temp intermediary to avoid changing the value of t
he working.dataset

          temp <- tk_augment_timeseries_signature(temp)
          #Adds extra variables to help the machine learning catch all tim
e related effects

          temp <- temp %>% mutate_if(is.character,as.factor) #Can't operat
e with character columns, set those to factors

          temp <- temp %>% mutate_if(is.ordered, ~ as.character(.) %>% as.
factor)
          #The augment time series adds ordered variables, which also need
to be converted

          as.h2o(temp)
          #Removes data that should not be given to the model, including t
he extra rows because those witll never be complete

        }, pos = .GlobalEnv) #Specify global environment to call them with
different functions later


        assign(paste0("Dataset.split.model.",i),{
          #Splitting the base data into training and testing randomly to a
void the model fitting better to a particular period

          h2o.splitFrame(data = eval(as.name(paste0("Dataset.model.",i))),
ratios = 0.8)
          #Use 20% for test set as that is a frequently used standard in t
he h2o documentation

        }, pos = .GlobalEnv) #Specify global environment to call them with
different functions later
```

```
        assign(paste0("Prediction.set.",i),{ #The set with the actual futu
re dates that will generate predictions
          period.matches <- as.logical({variable.lag.identifier>=i}+{names
(working.dataset) %in% Non.lagg.variables}+
          {names(working.dataset) %in% User.prediction.variable})
          #The prediction set must match the training and test sets

          temp <- working.dataset[Prediction.rows[1:i],period.matches]

          temp <- tk_augment_timeseries_signature(temp)
          #Adds extra variables to help the machine learning catch all tim
e related effects

          temp <- temp %>% mutate_if(is.character,as.factor) #Can't operat
e with character columns, set those to factors

          temp <- temp %>% mutate_if(is.ordered, ~ as.character(.) %>% as.
factor)
          #The augment time series adds ordered variables, which also need
to be converted

          as.h2o(temp)

        }, pos = .GlobalEnv)


        assign(paste0("Dataset.train.model.",i),{ #The training set for mo
del building

          eval(parse(text = paste0("Dataset.split.model.",i,"[[1]]"), "ind
ex")) #Extract the 1st split from the previous object

        }, pos = .GlobalEnv) #Specify global environment to call them with
different functions later



        assign(paste0("Dataset.test.model.",i),{ #The test set for model b
uilding

          eval(parse(text = paste0("Dataset.split.model.",i,"[[2]]"), "ind
ex")) #Extract the 2nd split from the previous object

        }, pos = .GlobalEnv) #Specify global environment to call them with
different functions later


        assign(paste0("H2o.model.",i),{
          #This is all the code for building the models. Kept simple to us
e the automatic defaults from h2o.
          #The most important metrics here are the manually set stopping m
etrics
          h2o.automl(y = User.prediction.variable, training_frame = eval(a
s.name(paste0("Dataset.train.model.",i))),
```

```
                    leaderboard_frame =  eval(as.name(paste0("Dataset.tes
t.model.",i))), max_runtime_secs = max.runtime)
        }, pos = .GlobalEnv)


        assign(paste0("H2o.model.",i,".performance"),{
          #Store performance metrics of the leader in a seperate frame to
facilitate evaluation
          h2o.performance(eval(parse(text=paste0("H2o.model.",i,"@leader")
,"index")))
        }, pos = .GlobalEnv)


        assign(paste0("H2o.model.",i,".important.variables"),{
          #Store the most important variables of each leading model for pr
oviding feedback to the user
          (eval(parse(text=paste0("H2o.model.",i,"@leader@model$variable_i
mportances"),"index")))
        }, pos = .GlobalEnv)

        assign(paste0("Model.",i,".predictions"),{
          #This makes a single column with the same length as the expanded
dataset holding all of the predictions from each model
          Lag.NAs <- setNames(as.data.frame(rep(NA, Lags.amounts)), c("pre
dict"))
          #There will always be a number of NA's euqal to the number of la
gs at the start of each dataset.
          base.pred <- as.data.frame(h2o.predict(eval(as.name(paste0("H2o.
model.",i))),eval(as.name(paste0("Dataset.model.",i)))))
          #The prediction model is applied to the actually observed dates
          actual.pred <- as.data.frame(h2o.predict(eval(as.name(paste0("H2
o.model.",i))),eval(as.name(paste0("Prediction.set.",i)))))
          #Model applied to the future dates to generate the actual predic
tions
          End.NAs <- setNames(as.data.frame(rep(NA, (Gap.actual+Prediction
.periods.actual-i))), c("predict"))
          #Each model has a number of NAs at the end equal to the perids t
hey can't cover
          rbind(Lag.NAs,base.pred,actual.pred,End.NAs)
        }, pos = .GlobalEnv)


        extended.dataset <- add_column(extended.dataset, temp.name = eval(
parse(text=paste0("Model.",i,".predictions[,1]"), "index"))
                                       ,.before = User.prediction.variable
)
        #Here the non-global object is updated with each models prediction
s in a new column
        names(extended.dataset)[names(extended.dataset) == "temp.name"] <-
paste0("Predictions based on ", i, " periods or older data")
        #The column is renamed

        #Combining all the different models to a single prediction column
        if(i==(1+Gap.actual)) {extended.dataset[,"Predictions"] <- eval(as
```

```
.name(paste0("Model.",i,".predictions"))))}
        #Because the first model has access to the most data it should def
inetly be the best, as a time period is removed the
        #models should become progressively worse
        {extended.dataset$Predictions[(Prediction.rows[i])] <- as.numeric(
tail(actual.pred, n=1))}
        #After the first model the single new prediction is added each tim
e

        #Filling out the evaluation metrics
        evaluation.metrics[i-Gap.actual,"Prediction model"] <- paste0(
          "Predictions based on ", i, " periods or older data")
        evaluation.metrics[i-Gap.actual,"Apx. accuracy (R^2)"] <- eval(
          parse(text=paste0("H2o.model.",i,".performance@metrics$r2"),"ind
ex"))
        evaluation.metrics[i-Gap.actual,"Mean Squared Error"] <- eval(
          parse(text=paste0("H2o.model.",i,".performance@metrics$MSE"),"in
dex"))
        evaluation.metrics[i-Gap.actual,"Root Mean Squared Error"] <- eval
(
          parse(text=paste0("H2o.model.",i,".performance@metrics$RMSE"),"i
ndex"))
        evaluation.metrics[i-Gap.actual,"Mean Absolute Error"] <- eval(
          parse(text=paste0("H2o.model.",i,".performance@metrics$mae"),"in
dex"))



      } #End of for loop
    }) #End of with progress function


    Dataset.base.3 <<- extended.dataset #After the for loop the global obj
ect is updated to include all predictions from all models

    Evaluation.metrics.data <<- evaluation.metrics #Creating the global ob
ject that will be rendered to the user

  })})


  output$Prediction.variable.plot.3 <- renderPlot({ try({
    ggplot(data = datasetInput.3$base(), aes(x=Date, y=eval(as.name(User.p
rediction.variable)))) +
      theme_economist() + geom_line(lwd=1) + geom_point(lwd=1) + {if("Pred
ictions" %in% colnames(datasetInput.3$base()))
        geom_line(aes(x=Date, y=Predictions), color="red3", size=1)} +
      labs(y=NULL, title = paste0(User.prediction.variable, " for all date
s. When completed, predictions are shown in red"))
  })})

  output$Plot.click.info.3 <- renderText({
    paste0("Click anywhere in the plot to show the values of that point","
\nDate = ", {if (is.null(input$Plot.click.3$x)) {NULL}
```

```
      else {as.Date(input$Plot.click.3$x)}}, "\n",User.prediction.variable
," = ", input$Plot.click.3$y)
  })


  #---------- Creating tables for the evaluation tab -------------------


  current.Dataset.4 <- reactive({datasetInput.3$base()})

  #Creating a reactive object so it can update with new values
  Evaluation.metrics.reactive <- eventReactive(input$Confirm.button.3,{try
({
    Evaluation.metrics.data
  })})


  #Creating table with the model important variables
  Important.variables.reactive <- eventReactive(input$Variable.names.4,{tr
y({
    i <- parse_number(input$Variable.names.4)
    as_tibble(eval(parse(text=paste0("H2o.model.",i,".important.variables[
,c(\"variable\", \"percentage\")]")),"index")))
  })})


  #Creating table with the model leaderboard for selected model
  Model.leaderboard.reactive <- eventReactive(input$Variable.names.4,{try(
{
    i <- parse_number(input$Variable.names.4)
    as_tibble(eval(parse(text=paste0("H2o.model.",i,"@leaderboard[,1]"),"i
ndex")))
  })})


  #The output object rendered as a table
  output$Base.dataset.table.4 = DT::renderDataTable(current.Dataset.4())

  #Evaluation metrics tble
  output$Evaluation.metrics.table = DT::renderDataTable(Evaluation.metrics
.reactive())

  #Important variables table
  output$Important.variables.table = DT::renderDataTable(Important.variabl
es.reactive())

  #Leaderboard table
  output$Model.leaderboard.table = DT::renderDataTable(Model.leaderboard.r
eactive())


  output$Prediction.variable.plot.4 <- renderPlot({ try({
    ggplot(data = datasetInput.4$base(), aes(x=Date, y=eval(as.name(User.p
rediction.variable)))) +
```

```
      theme_economist() + geom_line(lwd=1) + geom_point(lwd=1) + {if(input
$Variable.names.4 %in% colnames(datasetInput.4$base()))
        geom_line(aes(x=Date, y=eval(as.name(input$Variable.names.4))), co
lor="red3", size=1)} +
      labs(y=NULL, title = paste0(User.prediction.variable, " for all date
s. When completed, predictions are shown in red"))
  })})

  output$Plot.click.info.4 <- renderText({
    paste0("Click anywhere in the plot to show the values of that point","
\nDate = ", {if (is.null(input$Plot.click.4$x)) {NULL}
      else {as.Date(input$Plot.click.4$x)}}, "\n",User.prediction.variable
," = ", input$Plot.click.4$y)
  })


  #---------- Updating varaible lists for the datasets ------------------
-


  #Creating list of variable names for the current dataset
  observe({

    # Can also set the label and select items
    updateSelectInput(session, "Variable.names.1",
                      label = "Select variable(s) in the dataset",
                      choices = variable.names(current.Dataset.1()),
                      selected = NULL
    )})

  #Creating list of variable names for the current dataset in tab 2
  observe({

    x <- input$Base.dataset.1

    # Can use character(0) to remove all choices
    if (is.null(x))
      x <- character(0)

    # Can also set the label and select items
    updateSelectInput(session, "Variable.names.2",
                      label = "Select variable(s) in the dataset",
                      choices = variable.names(current.Dataset.2()),
                      selected = NULL
    )})


  #Creating list of variable names for the current dataset in tab 3
  observe({

    x <- input$Base.dataset.1

    # Can use character(0) to remove all choices
    if (is.null(x))
```

```
      x <- character(0)

    # Can also set the label and select items
    updateSelectInput(session, "Variable.names.4",
                      label = "Select the model you want to evaluate",
                      choices = variable.names({current.Dataset.4() %>% se
lect(contains("periods or older data"))}),
                      selected = NULL
    )})


  #Allows the user to download the complete dataset
  output$Download2 <- downloadHandler("thename.csv", try({
    content = function(filename){
      working.dataset <- as.data.frame(lapply(Dataset.base.2, function(x)
unlist(x)))
      working.dataset <- working.dataset[1:nrow(Dataset.base.2),]
      write.csv(x=working.dataset, file=filename, row.names = FALSE)
    }
  }))


  #Allows the user to download the data with all predictions
  output$Download3 <- downloadHandler("thename.csv", try({
    content = function(filename){

      req(input$Rename.delete.input.2, cancelOutput = TRUE)
      req(!is.na(as.integer(input$Rename.delete.input.2)), cancelOutput =
TRUE)


      working.dataset <- as.data.frame(lapply(Dataset.base.3, function(x)
unlist(x)))
      #Unlist to prevent breakdown from extra information in the variables
      working.dataset <- working.dataset[1:nrow(Dataset.base.3),]
      #The unlist pushes the extra infromation to the bottom of the table,
so it can be avoided
      working.dataset <- sample_n(working.dataset, size = as.integer(input
$Rename.delete.input.2))
      write.csv(x=working.dataset, file=filename, row.names = FALSE)
    }
  }))


  #Allows the user to download the data with all predictions
  output$Download4 <- downloadHandler("thename.csv", try({
    content = function(filename){
      working.dataset <- as.data.frame(lapply(Dataset.base.3, function(x)
unlist(x)))
      #Unlist to prevent breakdown from extra information in the variables
      working.dataset <- working.dataset[1:nrow(Dataset.base.3),]
      #The unlist pushes the extra infromation to the bottom of the table,
so it can be avoided
      write.csv(x=working.dataset, file=filename, row.names = FALSE)
```

```
    }
  }))



  } #End of server function

shinyApp(ui = ui, server = server)
```