Report: Applying the CRISP-DM Data Science Methodology to Sales Volume Forecasting and Budgeting Problems

Matthias Hofmaier (11944050)

August 4, 2023

Abstract

Reliable forecasts of a company's sales volume can be of massive advantage in budgeting and strategic planning. Traditional methods to forecast sales often rely on univariate moving average models. In the last decade, machine learning methods, which can incorporate information over various dimensions gained a lot of popularity. Those models allow the inclusion of variables from annual financial statements, including balance sheets and profit and loss statements, that potentially increase the prediction performance. But dealing with this data can be challenging. Especially for people who are from different domains and are not used to data science workflows. The Cross Industry Standard Process for Data Mining (CRISP-DM) can help to perform data science projects of this kind in a well-defined way. Therefore the goal of this project is to create a guideline project for students in economics that showcases how the CRISP-DM methodology can be applied to sales volume forecasting and budgeting problems. The project is carried out by comparing a univariate model (ARIMA) and a multivariate machine learning model (XGBoost) in the context of U.S. corporations in the S&P 500 from 2002 to 2022. The results show that despite extensive data preparation, the machine learning model is unable to exploit the exogenous variables and outperform the univariate model with the present dataset.

Main supervisor: Univ.-Prof. Dr. Walter Schwaiger, MBA[1]

Co-supervisor: Univ.-Prof. Dr. Allan Hanbury[2]

Selected domain-specific course: Project and Enterprise Financing (330.214), already completed.

1 Motivation

The forecasting of the sales volume is a substantial part of financial planning for any company in the world. Sales volume forecasting can help in cash flow and credit management, budgeting, production scheduling, and general decision-making within a business. A common way to do sales forecasting is to use univariate moving average models like ARIMA[1] on time series' of sales from the past. Another method is to employ machine learning algorithms like XGBoost[2] and apply them to multidimensional data. This method aims to use the information captured in the exogenous variables to achieve more reliable forecasts. For stock corporations, such variables can potentially be found in financial statements that have to be published annually. Especially data from balance sheets and profit and loss statements have the potential to improve the quality of forecasts as they also include information about the financial foundation of a particular company and its development over the previous years. But dealing with such data can be highly complicated and needs a well-defined process for understanding, preparing, modeling, and evaluation. Especially people who belong to different domains like economics are often overwhelmed when implementing data science workflows. A framework that can support people in the project implementation is the Cross Industry Standard Process for Data Mining (CRISP-DM)[3]. Therefore the goal of this project is to create a guideline project for students in economics that showcases how the CRISP-DM methodology can be applied to sales volume forecasting and budgeting problems. The project is carried out on the example of examining the differences between a one-dimensional moving average model (ARIMA) and a tree-based machine learning model (XGBoost) when performing such forecasts.

[1]Research Unit of Financial Enterprise Management, TU Wien
[2]Research Unit of Data Science, TU Wien

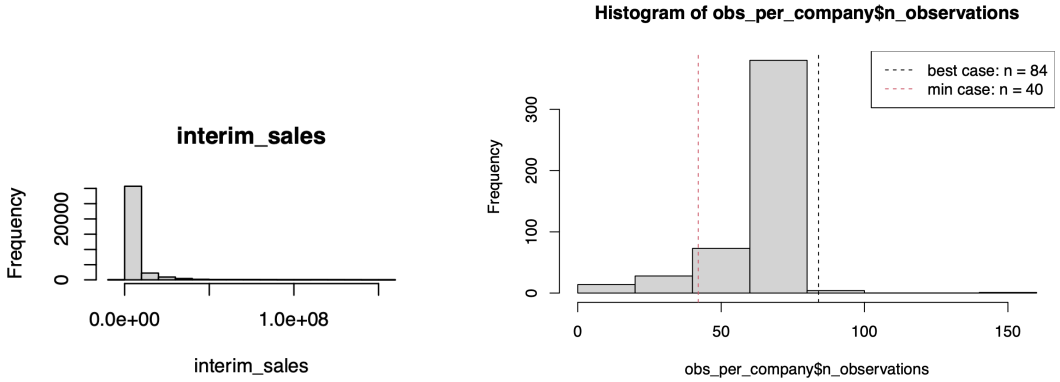
2 Methodology

Within this project, the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology will be followed. CRISP-DM is a widely-used process model for data science projects. It provides a structured approach with six main phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The model is iterative and cyclical in nature, allowing data scientists and analysts to revisit and refine previous stages as needed [3].
The first stage, Business Understanding, identifies the needs of a business and defines project goals and KPIs. As this project is part of a university course, the objectives and KPIs have already been defined in a proposal, so this stage is skipped. The first stage to be carried out is the Data Understanding stage (see Section 2.2). In this stage, data exploration techniques are used to describe the structure and assess the quality of the existing dataset. The findings of this stage are then passed on to the third CRISP-DM stage: Data Preparation (see Section 2.3). The objectives of this stage are to clean the dataset, deal with missing values, integrate and merge data from different tables, select data subsets, and derive new attributes. The next stage is Modeling (see Section 2.4). The goals of this stage are to apply different predictive or descriptive models to the prepared dataset. In the Evaluation stage (see Section 2.5), those models are then evaluated against the KPIs and metrics that have been defined in the Business Understanding stage. The final stage of the CRISP-DM methodology is the Deployment stage. Since the goal of this project is not to develop a service that can be deployed to production, this stage will be accomplished by documenting the project findings within this report [3].
In the following sections, the dataset which is used as the input for CRISP-DM together with the necessary transformations that have been applied is described. Followed by that, performed steps and findings of the Data Understanding, Data Preparation, Modeling, and Evaluation stages are elaborated. These sections are accompanied by a project documentation report[3] which includes code, outputs, and explanations for each of the performed steps.

2.1 Dataset

The dataset for this project stems from the Thomson Reuters Datastream database and includes the quarterly sales and variables from annual financial statements for S&P 500 stock corporations from 2002 to 2022. The financial statements consist of balance sheets with attributes describing the assets, liabilities, and equity as well as profit and loss statements which contain attributes describing the sales and expenses of a particular company. Datastream comes with an Excel interface but is not made to retrieve data in a clean way for multiple companies at once. The initial dataset consists of four Excel sheets for the interim sales variable (one for each quarter), one Excel sheet for the balance sheet variables, and two sheets for the variables of the profit and loss statements. The interim sales data is available on a quarterly frequency whereas the balance sheet and profit and loss variables are only available on a yearly frequency. The sales sheets contain rows for each year and quarter combination and columns for each respective company in the dataset. Since each sales sheet represents a different quarter, the values for the other quarters are missing. The balance sheet and profit and loss Excel sheets contain rows for each year and columns for each company and variable pair. For each of the quarterly sales sheets, we read the quarter and year from the row names and parse the company names from the column names. Finally, we merged all of the sheets into one data frame. For the balance sheet and profit and loss data, we similarly read the years from the row names and parsed company and variable names from the column header. The parsing is more challenging here since each column name consists of the structure `"COMPANY NAME - VARIABLE CATEGORY - COMPANY NAME - VARIABLE NAME"` and the spacing between the entities is not consistent throughout all entries. After merging the profit and loss sheets, we end up with one data frame containing the quarterly interim sales, one data frame containing the yearly balance sheet variables, and one data frame containing the yearly profit and loss attributes for each company in the dataset. The code that was used to achieve the aforementioned data transformations can be found in Section 1 of the project documentation report.

[3] See `project-documentation.pdf`



(a) Distribution of target variable interim sales.

(b) Distributions of the number of observations per interim sales time series.

Figure 1: Visualizations for the interim sales data frame.

2.2 Data Understanding

With the transformations we applied during the introductory step, we are now able to perform the first CRISP-DM stage: Data Understanding. Within this stage, we collect general information like the number of records and variables, visualize distributions, examine if there are gaps or duplicates in the times series' and finally assess the data quality by displaying the absolute and relative number of missing values for each variable in the available data frames. For the balance sheet and profit and loss data frames, we furthermore perform a correlation analysis by plotting the pairwise ranked cross-correlations between the variables.
By performing the aforementioned steps, we had the following core findings:

Sales Data:

- The sales data frame consists of 34841 rows with four variables for 500 distinct companies from 2002 to 2022, which translates to a time range of 21 years. Since the data is on a quarterly frequency, a time series of a particular company in the best case consists of 84 observations.
- The distribution of the target variable `interim_sales` (depicted in Figure 1 (a)) is highly right-skewed and has to be log-transformed to achieve a more symmetrical distribution.
- The distribution of the number of observations for each company (depicted in Figure 1 (b)) shows that the time series are of varying length. With an equally sized training and test dataset that spans over five years, we need at least 40 observations for a company to be included in the dataset. We can also see that there are time series that have more than 84 observations and contain duplicates, which have to be eliminated during data preparation.
- By measuring the distance between the points in each time series, we find 69 companies that have gaps in the data and have to be interpolated during data preparation. Except for those entries, we can not observe any missing values for any of the variables.

Balance Sheet Data:

- The balance sheet data frame consists of 10563 rows with 30 variables for 503 distinct companies from 2002 to 2022, which also translates to a time range of 21 years. Except for the company name, all of the variables are of numeric type. In contrast to the sales data, in this case, the time series are on a yearly frequency.
- Many of the balance sheet variables have highly skewed distributions that will be log-transformed during data preparation.

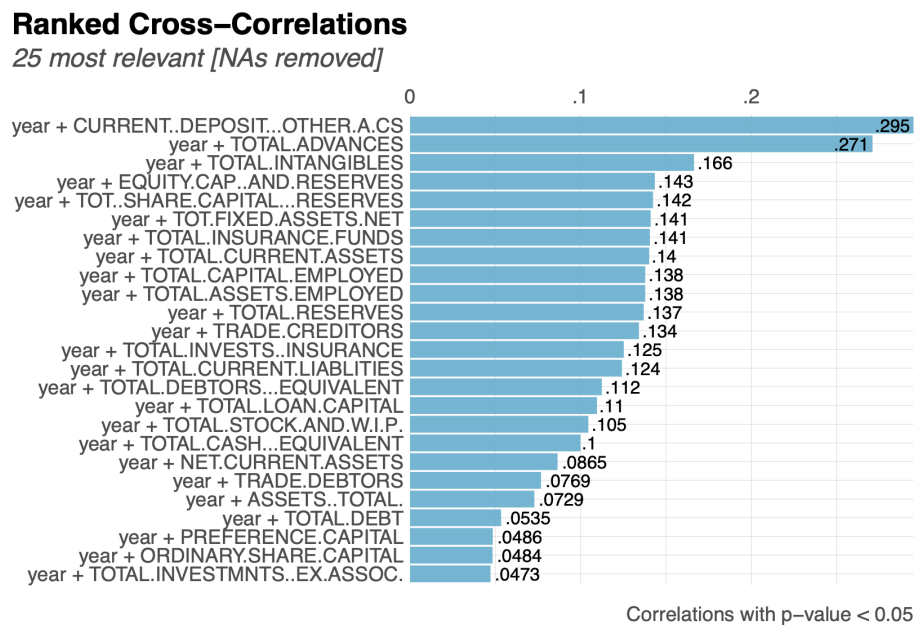


Figure 2: 25 most relevant ranked cross-correlations for the balance sheet data frame.

- In contrast to the sales data, the balance sheet time series do not contain gaps or have duplicates. Each time series has exactly 21 observations, but except for company and year, all variables have some missing values. The variable `TOTAL INSURANCE FUNDS` for example, has missing values for over 95% of the observations. During data preparation, we will remove all variables where more than 20% of the values are missing. This affects nine variables.
- A correlation analysis shows that the 25 most correlated variable pairs all include the `year` variable (see Figure 2). The highest correlation is between `year` and `CURRENT, DEPOSIT & OTHER A/CS` and has a value of 0.295. When excluding `year` from the analysis, no significant correlations between the other variables can be observed.

Profit and Loss Data:

- The profit and loss data frame consists of 10563 rows with 42 variables for 503 distinct companies from 2002 to 2022. As for the balance sheet data, this data frame is on a yearly frequency with a time range of 21 years. Also for this data frame, all variables except the company name are of numeric type.
- Also for this data frame, we can observe many right-skewed variables. Log transformation will be employed to "symmetrize" the distributions of those variables.
- Similar to the balance sheet data frame, we have 21 observations for each respective company time series with no duplicates. The variable `NET INTEREST INCOME` has over 95% missing values and similarly to the other data frame, we will remove all variables where more than 20% of the values are missing. For the profit and loss data frame, there are 10 variables that will be removed.
- The result of a correlation analysis for this data frame, depicted in Figure 3 looks different than for the balance sheet data and shows strong correlations between the variables. This makes sense since the values of a profit and loss statement are all highly dependent on each other. The variables `EBIT` and `EBITDA` only have the difference that the `EBIT` describes the earnings before interests and taxes and the `EBITDA` furthermore excludes deprecation and amortization from the earnings. This leads to the maximum correlation value of 0.973. It is interesting to see, that the 25 highest correlated pairs do not include the `year` variable, because the other variables already have such high correlations. Since highly correlated variables can lead to instability when modeling the data, we will perform two strategies to remove the correlations in the data preparation stage:

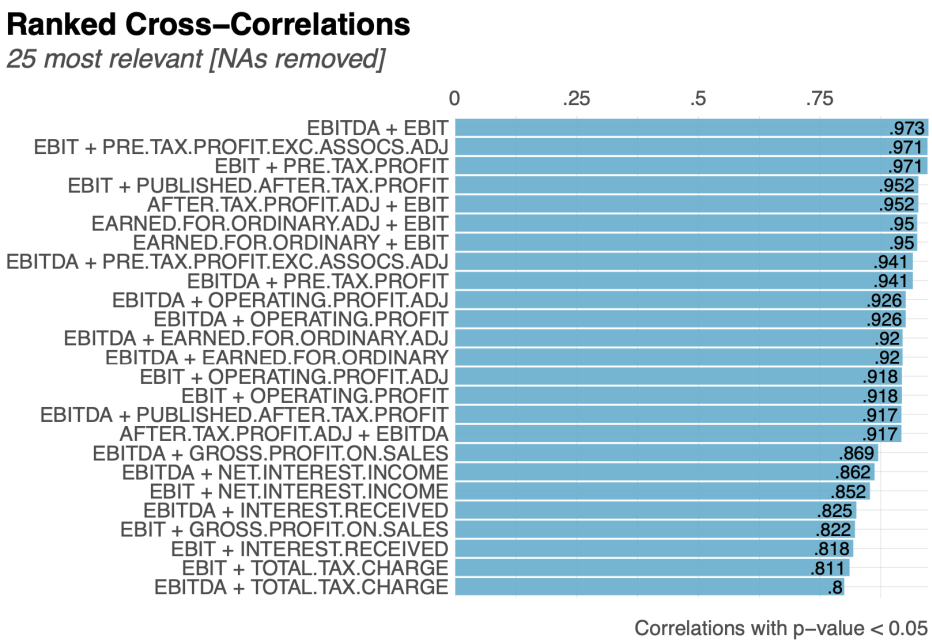


Figure 3: 25 most relevant ranked cross-correlations for the profit and loss data frame.

(1) We will employ principal component analysis (PCA) to reduce the dimensionality of the profit and loss variables and thereby remove the correlation and (2) only keep the variables for which we have the most data and drop all other variables that are highly correlated to this variable.

General Findings:

- We observed that the company names in the sales data frame follow a different naming scheme than the company names in the balance sheet and profit and loss data frames (e.g., APPLE and APPLE INC) and we will have to take care of this when joining all data frames together.
- The variable names for the balance sheet and profit loss data frames are capitalized and contain white spaces and special characters. The variable names should be transformed to lowercase, and white spaces and special characters should be replaced or removed during data preparation.

All of the previously stated findings are now input into the Data Preparation stage, in which the necessary data preparation and transformation steps will be performed. A more extensive elaboration and all other findings we gained during the Data Understanding stage can be found in Section 2 of the project documentation report. Note that since we are dealing with 500 distinct companies with many variables, a visualization of all of those time series' was not possible.

2.3 Data Preparation

We will now use the findings gained from the Data Understanding stage to enter the Data Preparation stage of CRISP-DM. The goals of this stage are to log-transform the target variable and other skewed, but positive variables, remove duplicated data points and exclude companies with not enough observations, interpolate time series with gaps or missing values, remove variables with more than 20% of missing values, transform the variable names, join the data frames in a fuzzy manner due to the varying company naming schemes, enrich the data with information about their industry sectors, perform two strategies to remove the correlation between variables, and finally perform a train test split, by using the last five years of each time series for testing.
In the following sections, we will describe the main transformations we applied to the data tables:

Sales Data:

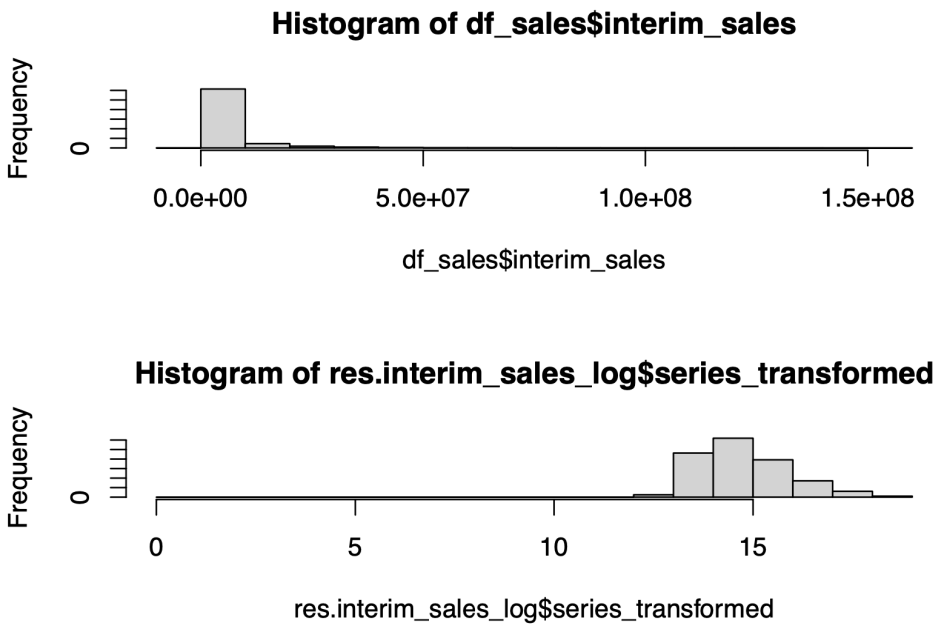


Figure 4: Histogram of target variable before and after log transformation.

- We log-transform the target variable `interim_sales`. Since the time series' also contain negative values, we add a constant of $min(interim\_sales) + 1 = 393001$ to the target variable to ensure that all values are positive. The result of the log transformation is indeed a more symmetrical distribution (see Figure 4). The addition of the constant before the log transformation also introduced an outlier point ($log(1)$) which we replace with an interpolated value.
- In the second step, we remove 146 duplicated data points and 37 companies with less than 40 observations from the data frame. After the removal of some companies, the data frame still contains 64 time series' with gaps. We employ spline interpolation to fill those gaps without removing companies.

Balance Sheet Data:

- In the first step, we transform the variable names of this data frame to lowercase, replace white spaces with underscores and remove or replace special characters. Followed by that we remove the nine variables with over 20% of missing values.
- The next step is to log-transform right-skewed variables. Since we have to deal with 21 variables, we are not able to perform an in-depth investigation of each variable we want to transform. Thus, we visually select only highly right-skewed variables that are positive such that we do not have to add a constant.
- Since it is not possible to interpolate variables of companies, where not at least two observations are available, we remove companies where more than 50% of values for any variable are missing. This step excludes 128 companies from the dataset. Now, we are able to perform the spline interpolation as planned and end up with no missing values for any of the variables of the balance sheet data frame.

Profit and Loss Data:

- Similarly to the balance sheet data frame, we start with the transformation of the variable names. Followed by that, we remove the 10 variables with over 20% of missing values, log-transform positive right-skewed variables, exclude companies where more than 50% of values for any variable are missing (147 companies in this case), and finally perform spline interpolation. This results in no missing values for any variable in the profit and loss data frame.

Joint Preparations:

- The joint preparation begins with a fuzzy join of the company names of the sales data frame with the company names of the balance sheet and profit and loss data frames. The fuzzy join is performed by using the Jaro-Winkler string distance, such that each company name pair that has a distance smaller than 0.3 is considered a match. This method works for most of the company names but also results in companies that have multiple, wrong, or no matches. After some manual matching adjustments, we end up with 313 unique matched companies in the dataset.
- Since we want to predict the interim sales for an upcoming year by using features from the actual year (e.g., with features from 2002 we want to predict the sales for 2003), we shift our target variable one year backward (i.e, the year of a data point in the sales data frame is decremented by one).
- The next step is to perform a correlation analysis on the joined data frame that includes all variables. In Figure 5, we can observe many highly correlated variable pairs. This is expected but since this can introduce errors in our multivariate modeling approach, we remove those correlations with two strategies: 1) Perform variable selection by removing the first variable in a highly correlated pair (above an absolute correlation value of 0.6) and 2) Perform dimensionality reduction by employing principal component analysis (PCA). The first strategy removes 30 variables (26 remaining) from the data frame and for the second strategy, we select the first 20 components to capture 95% of the variance and end up with 27 remaining variables. The dataset that uses a variable selection approach now has no variable pairs with an absolute correlation value of 0.6. The PCA dataset does not show any correlated variable pairs.
- Outside of the used R notebooks, we enrich the companies in the dataset with information about their respective industry sector. This is done by linking the companies to the open knowledge base WikiData[4] in OpenRefine.
- The final step of the Data Preparation stage is to conduct a train test split of the data. As mentioned earlier, we use the last five years of each time series for testing purposes. Through previous processing steps, we ensured that the training part of the time series is at least also five and at most 15 years long. Over all time series, this results in a train set size of 71.8% and a test set size of 28.2%.

The datasets that were created by executing the aforementioned data preparation steps, can now be used for modeling. A more detailed elaboration of the Data Preparation stage together with the code can be found in Section 3 of the project documentation report.

2.4 Modeling

With the newly created datasets, we now enter the Modeling stage of CRISP-DM. In the following sections, we will first of all perform naive forecasts, that use the last observation of the training dataset as the forecasting value. This simple method will act as a baseline to evaluate the more sophisticated approaches. Followed by that, we will fit ARIMA models for each particular company time series. Finally, we will train two XGBoost models, one using the dataset with variable selection and one using the dataset that we generated by using principal components, to forecast the interim sales variable. The code that was used within this stage can be found in Section 4 of the project documentation report.

2.4.1 Naive Forecasting

The naive forecast that acts as a baseline for the subsequent models is implemented by simply taking the last value of each time series in the train set and using this value as a forecast for all time points in the forecasting horizon of five years.

[4] www.wikidata.org

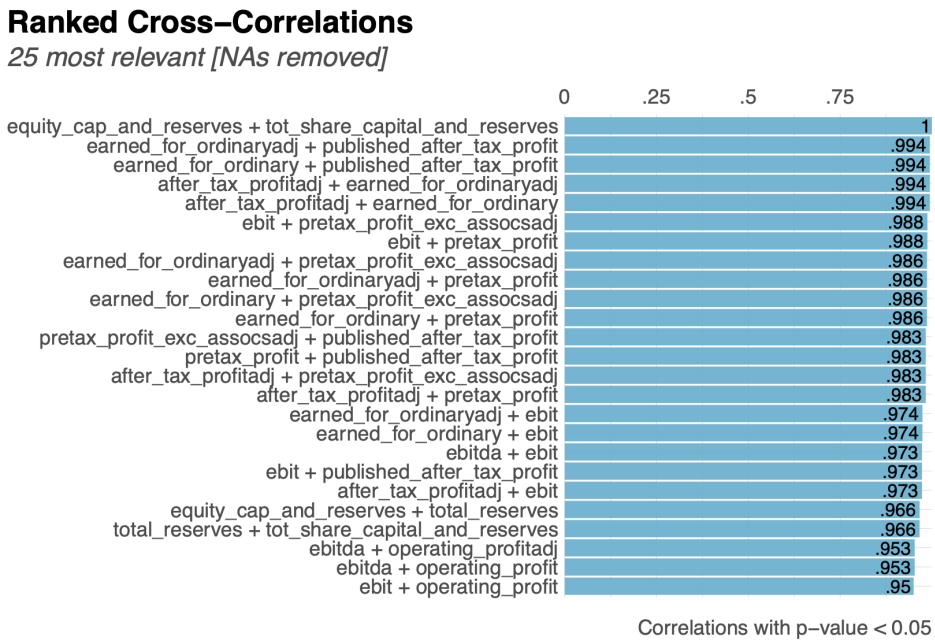
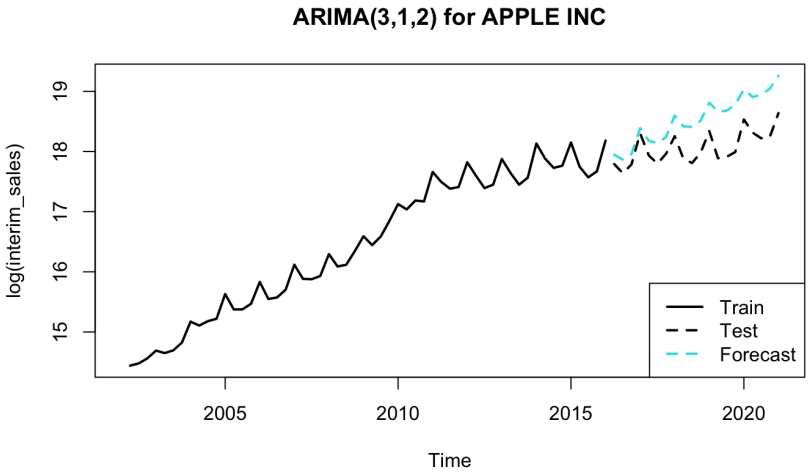


Figure 5: 25 most relevant ranked cross-correlations for the joined data frame.

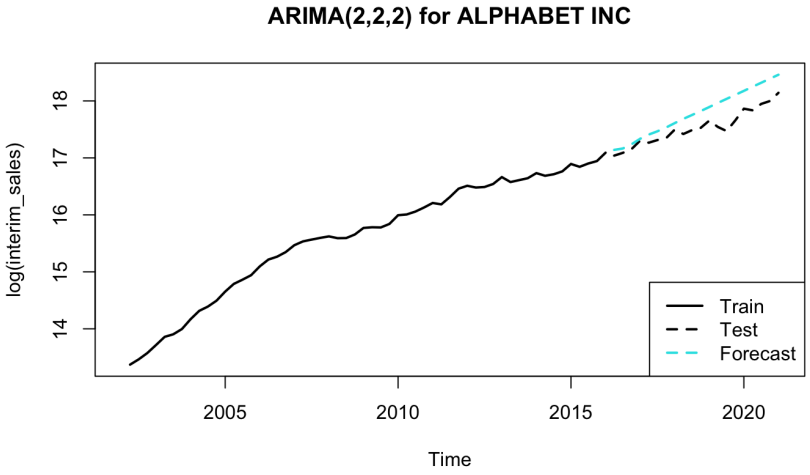
2.4.2 ARIMA

ARIMA (Autoregressive Integrated Moving Average) is a one-dimensional mathematical model for forecasting future values in time series data. It involves three key components: autoregression (AR), differencing (I), and moving average (MA). The autoregressive component examines the relationship between the current value of a variable and its previous values. It assumes that the current value can be explained by a linear combination of past observations. The order of autoregression (p) determines the number of previous observations that are considered in the model. The differencing component (I) is employed to transform the time series into a stationary series, i.e., a time series that does not have seasonality and trend. Seasonality and trend are eliminated by computing the differences between consecutive observations. The order of differencing (d), specifies the number of differencing operations needed to achieve stationarity. The moving average component (MA) captures the short-term fluctuations or noise in the data. It considers the relationship between the current value and past forecast errors. The order of the moving average (q) specifies the number of previous forecast errors to include in the model [1].
To determine the appropriate values for the ARIMA parameters (p, d, q), the autocorrelation function (ACF) and partial autocorrelation function (PACF) are typically analyzed. As we deal with many time series' at once, we can not analyze the ACFs and PACFs by hand. Fortunately, there is a function in R (`auto.arima()`) which automatically finds the ARIMA parameters for a particular time series.
The first step of the ARIMA modeling is to load the data and fit an ARIMA model for three example companies (Apple, Alphabet, and Amazon). The training time series, the ARIMA forecasts, and the actual data for those three companies are visualized in Figure 6.

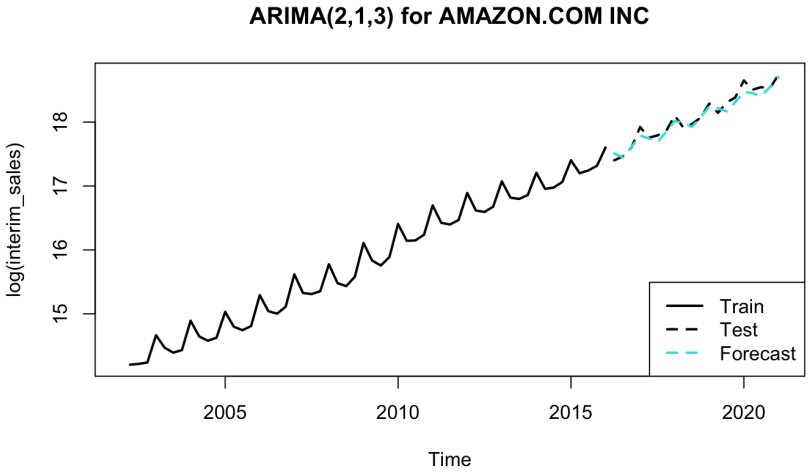
By looking at the forecasts for the example companies depicted in Figure 6, we can see that `auto.arima()` did a reasonable job and found different parameters for each of the time series. For Apple and Amazon, the automatically fitted ARIMA model uses first-order differencing (d=1). For Alphabet, the algorithm chose second-order differencing (d=2). A reason for this is the different structure of the time series, as it is also visible in the plots. Especially for Amazon the ARIMA model looks very promising. Here the forecasted values are almost identical to the values in the test data. For the Apple time series, ARIMA does a good job of capturing the seasonal pattern of the series but overestimates the slope of the trend. For Alphabet, the model captures almost no seasonal structure and also slightly overestimates the trend of the time series.
After fitting the models for the example companies, we then fitted the models for all the companies without further visual inspection.



(a) ARIMA(3,1,2) forecast for Apple.



(b) ARIMA(2,2,2) forecast for Alphabet.



(c) ARIMA(2,1,3) forecast for Amazon.

Figure 6: Visualizations of the ARIMA forecasts for three example companies.

2.4.3 XGBoost

XGBoost (eXtreme Gradient Boosting) is a multivariate machine learning model. This model will use the exogenous variables from the balance sheet and profit and loss statements to forecast the quarterly sales of particular companies. In contrast to the ARIMA section, where we fitted a separate model for each time series, we will only train one XGBoost model that is capable of generating forecasts for all companies. XGBoost is an ensemble model that combines multiple decision trees to create a forecasting model. It works by iteratively building a series of decision trees and then combining their predictions to obtain the final forecast. Each decision tree is trained to minimize the errors of the previous trees, resulting in a more accurate ensemble model [2]. However, it's important to note that the prediction quality of a machine learning model like XGBoost normally increases with the number of data points it can use for training. Since our dataset only consists of 15 training data points for each particular company, there is the possibility that XGBoost cannot show its full potential.
Because we previously created two separate datasets, one using a variable selection approach and one using principal components as variables, we will train a separate model for each of those. For each model, we will perform a parameter search. The parameter search uses five different values for each particular parameter and builds a grid of all of those parameter combinations. The quality of a parameter combination is evaluated by using a group 5-fold cross-validation approach. This means that the training dataset is divided into five equally sized folds based on the company names (such that a time series is not in multiple folds at the same time). The model is trained on four folds and validated on the remaining fold in each iteration, creating five separate evaluations. The final performance metric is the average of these five evaluations. The parameter combination that yields the best value for the performance metric (in this case the lowest Root Mean Squared Error (RMSE)) is then selected as the final parameter setting for the model. The final parameters found during the parameter search are shown in Table 1. Note that we also tested a version that did not use a 5-fold cross-validation approach. This version did result in lower errors for the training data but higher ones for the test data, which is a clear sign of overfitting.

Parameter \ Dataset	**Variable selection**	**PCA**
eta	0.3	0.4
max_depth	4	2
gamma	0	0
colsample_bytree	0.8	0.8
min_child_weight	1	1
subsample	0.5	0.625
nrounds	50	250
RMSE	**0.284**	0.309

Table 1: Hyperparameters with the lowest RMSE for XGBoost models using variable selection and PCA data.

In Table 1, we can observe slight differences in the found parameters between both datasets. The parameter tuning found an *eta* value of 0.3 for the variable selection dataset and a value of 0.4 for the PCA dataset. *Eta* controls how much information from a new tree will be used in the Boosting. If it is close to zero only a small piece of information from each new tree is used. If *eta* is set to 1 all information from the new tree is used. For *max_depth*, the parameter tuning chose a value of 4 for the variable selection and a value of 2 for the PCA data. As the name suggests, *max_depth* controls the maximum depth of the trees. Deeper trees have more terminal nodes and fit more data, but are also more prone to overfitting. *Gamma* specifies the minimum loss reduction to make a split within a tree and is kept at zero for both datasets. The *colsample_bytree* value of 0.8 is again similar and denotes the proportion of variables that will be used to construct a new tree. Also, the parameter *min_child_weight* is similar for both datasets with a value of 1. It defines the minimum sum of weights of all observations required in a child and is used to control overfitting. Higher values prevent a model from learning relations that might be highly specific to the particular sample selected for a tree. The parameter *subsample* is again different between the datasets (0.5 vs. 0.625) and denotes the propor-

tion of observations that are selected to build a new tree and can also be used to control overfitting. Furthermore, the *nrounds* parameter is five times bigger for the PCA than for the variable selection data and denotes the number of trees that are used within the model. One can summarize, that the found parameters represent an ensemble model that consists of more but less deep trees for the PCA dataset than for the variable selection dataset. By looking at the RMSEs, we can see that the error of the model that uses variable selection data is slightly lower than the error of the PCA model.
Similarly to the ARIMA modeling, we will visualize the produced forecasts for three example companies, shown in Figure 7 and 8.

When looking at the forecasts of the variable selection XGBoost model depicted in Figure 7, we can see that they look less promising than the ARIMA forecasts. For Apple, XGBoost does a good job of forecasting the mean of the first few years. After that, the forecast suddenly drops and stays constant. For Alphabet and Amazon, both the trend and the seasonality are not estimated correctly. A reason for that can be the few training observations for each particular company.

Also, the model using PCA data, depicted in Figure 8 shows less promising results than the ARIMA forecasts, but differs compared to the variable selection model. At least for Alphabet and Amazon, we can see that the forecasts are closer to the test data for the first years but suddenly show strange behavior afterward. From visual inspection, the PCA model seems to make better forecasts for Apple and Alphabet than the model using variable selection data, but the forecasts of both models look unnatural compared to the ARIMA forecasts.
In the next section a quantitative evaluation of all models with be conducted.

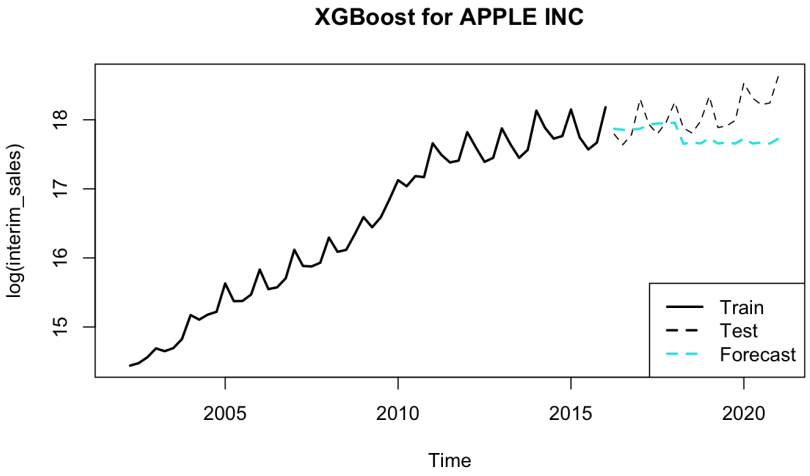
2.5 Evaluation

The Evaluation stage is the final CRISP-DM stage we will perform within this project. As the name suggests, the goal of this stage is to evaluate the models we produced in the previous stage. In the following sections, first of all, the employed evaluation metrics are elaborated, followed by an explanation of the performed significance tests and an analysis of the model performance across industry sectors. The code that was used within the Evaluation stage can be found in Section 5 of the project documentation report.

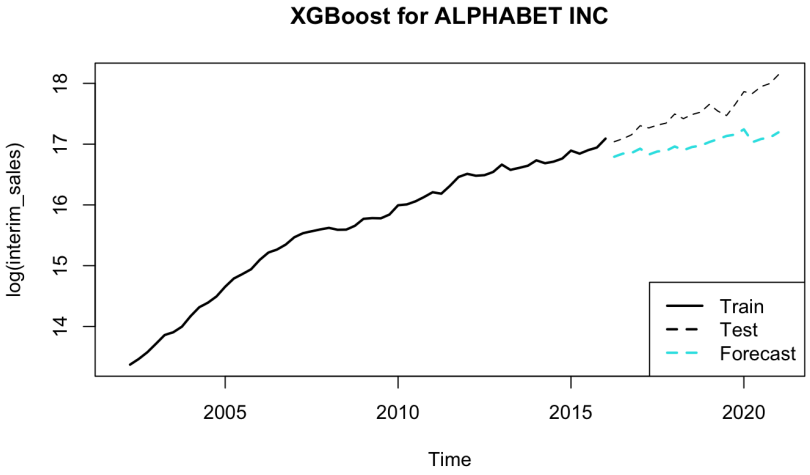
2.5.1 Evaluation Metrics

In our project, we employ three evaluation metrics to assess the performance of our models: Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Scaled Error (MASE). RMSE is a metric that calculates the average magnitude of the errors between our model's forecasts and the actual values. The RMSE penalizes larger errors more heavily than smaller ones, resulting in a single value that represents the overall accuracy of our models. The MAE also measures the average error between our model's forecasts and the actual values. But unlike RMSE, it treats all errors equally without giving more weight to larger errors. As a result, MAE provides a straightforward understanding of the accuracy of our model on the same scale as the target variable. Although the RMSE and MAE are the more well-known evaluation metrics, we will use the MASE as our main measure. The MASE is a metric used to evaluate the accuracy of a forecast model by measuring the relative performance of a forecasting method by comparing the mean absolute forecast errors to the mean absolute errors of a naive forecast. The MASE is scale-independent, and therefore suitable to compare the forecast performance across our different company time series' with varying scales [1].
All of the above evaluation metrics are calculated using the test set for each model on all of the time series' on the logarithmic and original scale of the target variable. The results are depicted in Table 2.

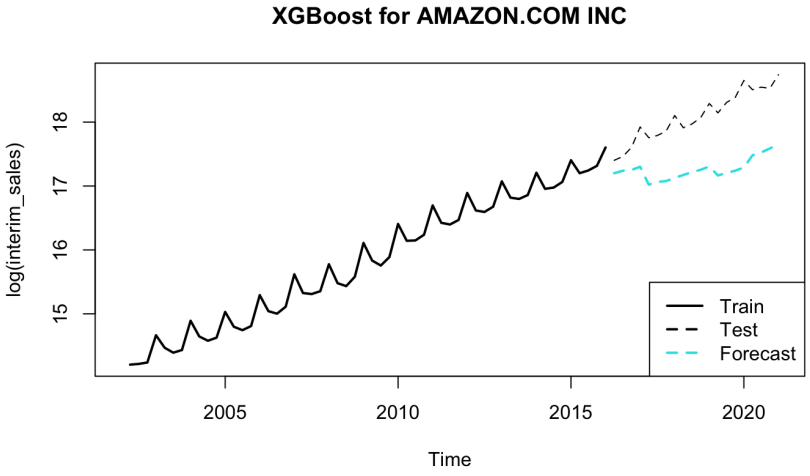
We can clearly see, that the ARIMA model has the lowest error across all metrics and scales and therefore outperforms the naive forecast as well as the XGBoost variants. It is also observable that both XGBoost models are not able to outperform the naive baseline. By looking at the MAE on the original scale, we can see that although ARIMA has the lowest errors, the model is still not sufficient to perform reliable forecasts of the target variable. The target variable ranges from -393,000 to 152,859,000 and thus a mean absolute error of 1,311,192 for ARIMA is considerably high and only marginally smaller than the baseline error. The next step will now be to perform significance tests to validate the results.



(a) XGBoost with variable selection forecast for Apple.

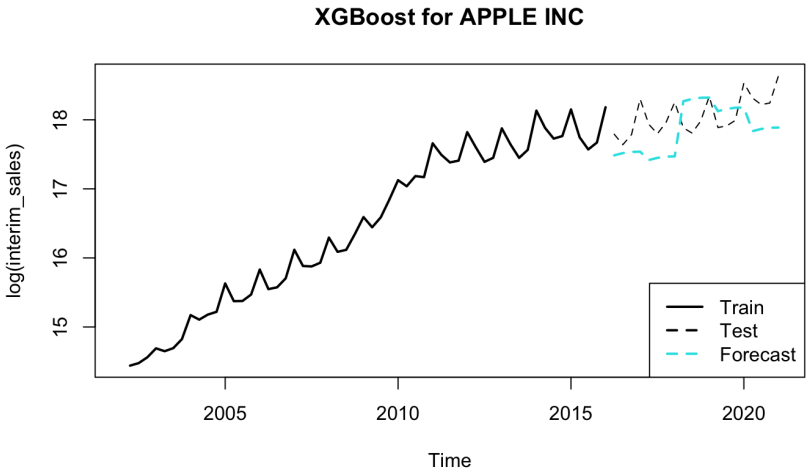


(b) XGBoost with variable selection forecast for Alphabet.

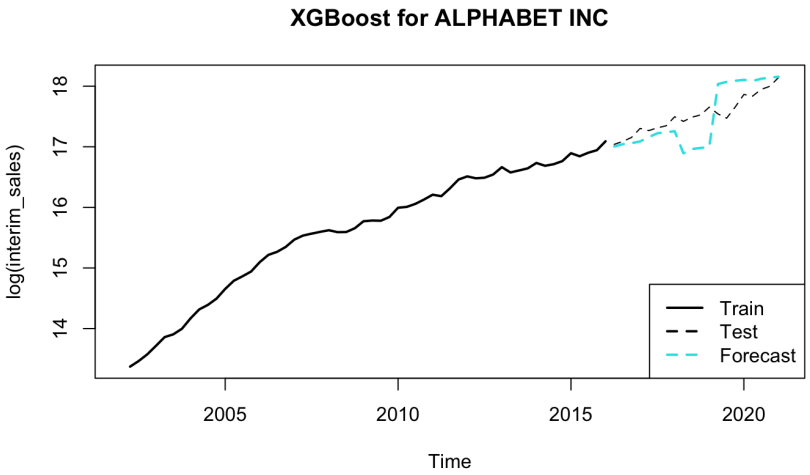


(c) XGBoost with variable selection forecast for Amazon.

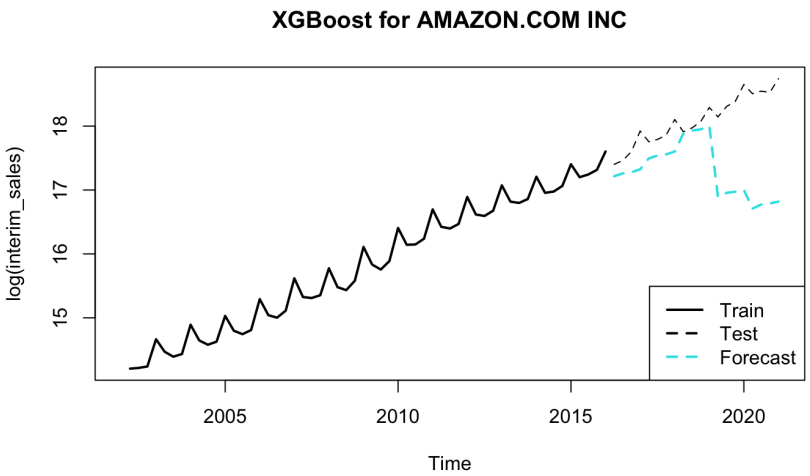
Figure 7: Visualizations of the XGBoost with variable selection forecasts for three example companies.



(a) XGBoost with PCA forecast for Apple.



(b) XGBoost with PCA forecast for Alphabet.



(c) XGBoost with PCA forecast for Amazon.

Figure 8: Visualizations of the XGBoost with PCA forecasts for three example companies.

model	scale	RMSE	MAE	MASE
naive forecast	log	0.3170303	0.2031872	0.6050440
ARIMA	log	**0.2915310**	**0.1789881**	**0.5329846**
XGBoost variable selection	log	0.3224325	0.2284652	0.6803160
XGBoost PCA	log	0.3428689	0.2357713	0.7020718
naive forecast	original	4780981	1405457	0.5817044
ARIMA	original	**4462916**	**1311192**	**0.5426892**
XGBoost variable selection	original	6520754	1957451	0.8101689
XGBoost PCA	original	6796391	2150901	0.8902361

Table 2: Evaluation metrics for all time series across different models and scales.

2.5.2 Significance Tests

We perform significance tests by using a Wilcoxon signed rank test with two paired samples. The Wilcoxon signed-rank test is especially useful when we want to compare two related groups, where the data does not meet the assumptions of parametric tests like the paired t-test (e.g., non-normal data or small sample sizes) [4]. We use the MASE metric within our significance tests.

X \ Y	naive forecast	ARIMA	XGBoost variable selection	XGBoost PCA
naive forecast	-	**9.748e-06**	0.9999	1
ARIMA	-	-	1	1

Table 3: P-values of the two-sample Wilcoxon signed rank tests with H_0 : MASE(X) $\leq$ MASE(Y) and H_1 : MASE(X) $>$ MASE(Y). The MASE is computed for each time series separately.

Table 3 shows the p-values of the two-sample Wilcoxon signed rank tests with the null hypothesis that the MASEs of X are less than or equal to the MASEs of Y and the alternative hypothesis that the MASEs of X is greater than the MASEs of Y. We see that we can only reject the null hypothesis for the test scenario where X = naive forecast and Y = ARIMA. For all other scenarios, we observe p-values of close to one or one and therefore definitely cannot reject the null hypothesis. This means that only ARIMA is able to outperform the naive forecast and the XGBoost models underperform the naive forecast, as it was also expectable from the results in Table 2. Of course, this also means that none of the XGBoost variants are able to outperform the ARIMA forecasts.

2.5.3 Industry Sector Analysis

The final step of the Evaluation stage is to compare the model performances across industry sectors. For that, we employ the industry sector information from WikiData we obtained during data preparation. For the comparison, we only include industries with at least five companies in our dataset. The data is grouped by industry and model and then aggregated by averaging the computed MASE for each of the companies. The results of this comparison are depicted in Figure 9.

We can see that the performance of the models differs between the industry sectors. For the telecommunications industry and the pharmaceutical industry, for example, all models have higher errors than the naive forecast. For the petroleum industry and financial services, on the other hand, we can observe that all models - even the XGBoost variants - outperform the naive forecast. It is also interesting to see, that for some industry sectors, the XGBoost with PCA data produces lower errors than XGBoost using the variable selection data. All of those findings indicate, that it can be beneficial to add the industry sector information to the training dataset or to train separate models for each industry sector.

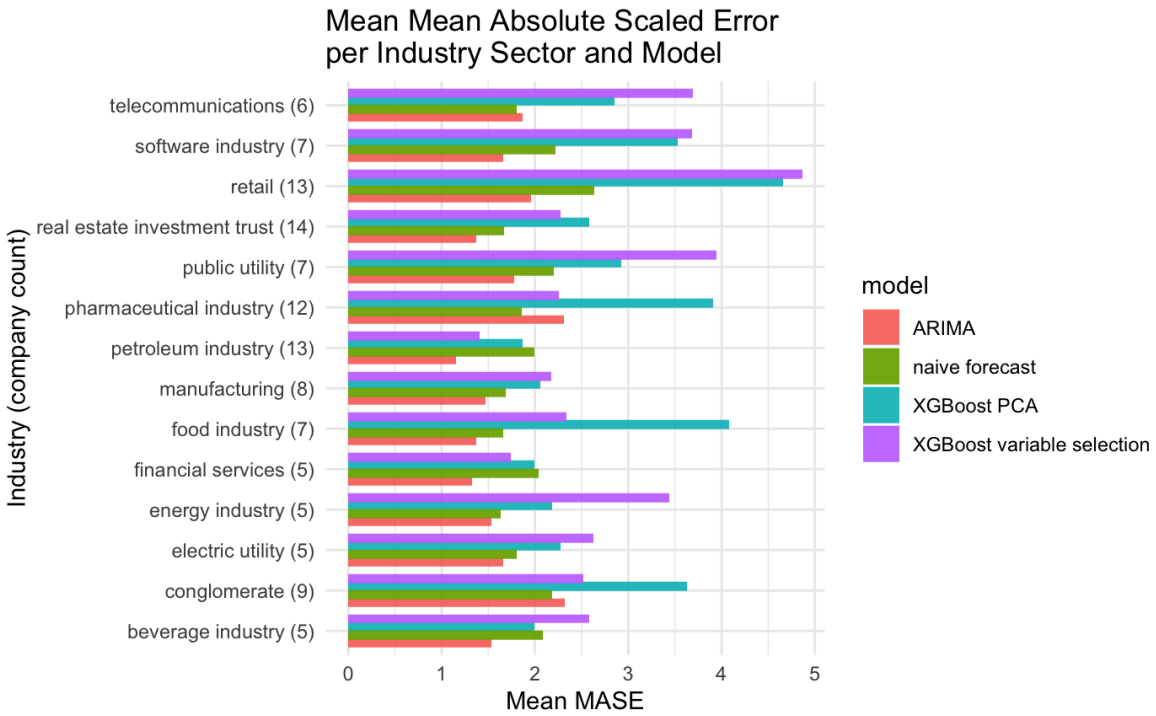


Figure 9: Comparison of model performances across industry sectors.

3 Discussion and Future Work

With the present project, we successfully showcased how the CRISP-DM data science methodology can be applied to sales volume forecasting and budgeting problems. We started with data export that ranged over multiple Excel sheets and was hardly usable for a data science project. After some initial transformations, we conducted the Data Understanding stage of CRISP-DM. The findings gained during Data Understanding were then input into the next stage Data Preparation. Within this stage, we extensively applied data transformations to the dataset. This stage again highlighted how expensive and time-consuming data preparation can be when conducting data science projects. After the Data Preparation stage, we ended up with datasets that could be used for the following CRISP-DM stage: Modeling. During Modeling, we implemented a naive baseline, a univariate ARIMA model, and two XGBoost models, which were using two different datasets. The Evaluation stage of CRISP-DM showed that despite the extensive data preparation efforts, the XGBoost models are unable to exploit the exogenous variables and outperform the baseline or the univariate ARIMA model with the present dataset. A possible reason for this is that the prediction quality of machine learning models like XGBoost normally increases with the number of data points that can be used for training the model. Since the used time series' were of rather small length (maximum 15 years for training), the model was potentially not able to capture all necessary patterns. A comparison of the model performance across industry sectors also revealed that there are differences in the errors between models and industries. This finding suggests including the industry sector information as a feature in the dataset or training separate models for each industry sector.

For further project iterations, it is recommended to enlarge the dataset by including more companies and widening the time range (e.g., to 50 years), such that the XGBoost models can benefit from a larger training dataset. It can also be beneficial to examine more sophisticated variants of ARIMA like SARIMA or SARIMAX, which can also exploit seasonal information or even exogenous variables for time series forecasting. Another potential improvement could be to combine the forecasts of multiple models, such as ARIMA and an XGBoost model trained on more data, through (weighted) averaging to achieve more reliable results.

References

[1] Rob Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 3rd edition, 2021.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[3] Rüdiger Wirth and Jochen Hipp. Crisp-dm: towards a standard process modell for data mining. 2000.

[4] Dirk Taeger and Sonja Kuhnt. *Statistical hypothesis testing with SAS and R*. Wiley, Chichester, West Sussex, 2014.