# Introduction

Finvia is an asset management firm based in Germany. Established in 2019 it is a growing, family-led company with now roughly 100 employees. To best allocate recourses in the marketing and sales department, Finvia needs to predict the number of leads arriving through their website and personal contact via telephone or office visit.

In marketing, a lead is any potential costumer, who comes into a first contact with the company and shows interest (Heinrich, 2020, S. 208). The subsequent steps are acquisition and retention (Kumar & Reinartz, 2018, S. 31). Both steps are recourse intensive. An accurate and secure prediction is therefore very helpful for any asset management firm.

The paper is structured in five chapters: Firstly, the data will be presented, broadly and with an exploratory data analysis. Afterwards data preparation includes the steps of data cleaning, wrangling, and merging. Then, the modeling will be explained and lastly our results, variable importance, and the performance metrices will be evaluated.

# Data Understanding

The data is divided into four separate tables: *Leads*, *Ads*, *WebsiteTraffic*, and *Macro* Data, each with a common *Date* column. The beginnings of the tables vary from the 1st of January 2020 (Macro) to the 14th of June 2021 (Ads). All tables last until the 31st of March 2023. The dependent variable *NextDayLeads* is stored in the *Leads* table.

In the table *Ads* numeric values concerning advertising expenditure (*Spend)* and conversion rate (*Impressions* and *Clicks*) are stored. Both variables are split based on platform and funnel. The project is partly based on the assumption that these values have a (lagging) influence on the number of leads.

The table *WebsiteTraffic* focuses on finvia.fo's website traffic, measuring the number of visitors and total time spent on the website. Again, both variables are assumed to have a positive influence on the number of leads by the company. All values until the 15th of August 2021 are equal to zero. Since the company was founded in 2019, a measurement error is assumed (FINVIA, 2023).

The *Macro* table stores macroeconomic data in 14 variables, like the DAX, the electricity and gold price. It only contains 824 variables in the span of 2958 days. This is because all values are only available for banking days.

# Exploratory Data Analysis

An exploratory data analysis was conducted to better understand the data and its structure. Graphs support this progress.

A provided graph showing *NextDayLeads* over time revealed some extreme outliers, for both the *manual* and the *website* type. Since the clear and timely assignment of leads is difficult (Heinrich, 2020), these outliers suggest measurement errors. Figure 1 shows the weekly sum of *NextDayLeads* after filtering the outliers. This filtering will be described in Chapter 3. Figure 1 clearly suggests that the number of leads is rising over time. This trend shows the increasing importance of this project.

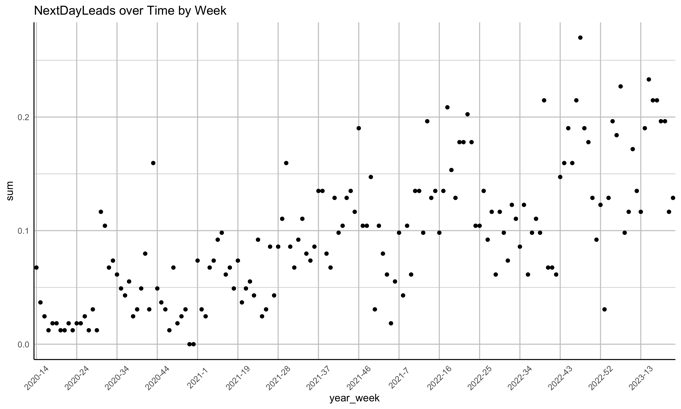


Figure 1: Leads over Time Aggregated by Week, after deleting Outliers.

The analysis of macro-economic data and its relation to leads has found little correlation between the variables. Two variables, the German Electricity price and the US Economy Uncertainty Index stand out. Figure 2 suggests a positive correlation between the German Electricity Price and leads and a negative correlation between the US Economy Uncertainty Index and leads. These correlations are based on the assumptions, that low electricity prices increase fiscal flexibility for consumers and that a low uncertainty gives consumers trust in the financial markets. On the other hand, the colors suggest that these correlations might rather be explained on the time component, which correlates with all three variables (see Figure 1).

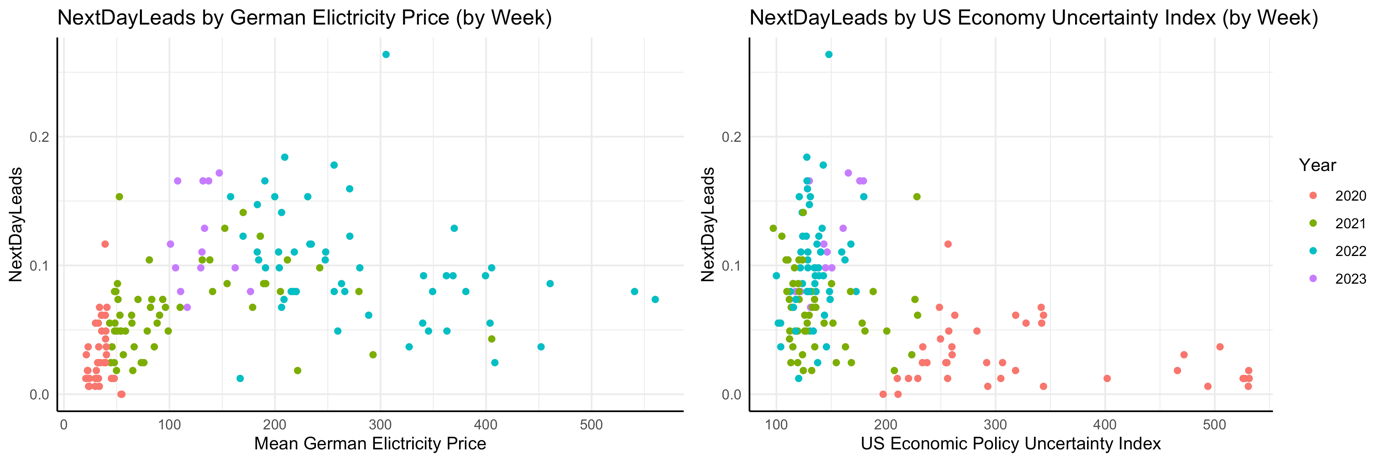


Figure 2: Leads by Week compared to Electricity Price and US Uncertainty Index.

Moreover, relations between the advertisement and traffic data have been examined. While correlations between *Visits* and *TimeSpent* and between *Clicks* and *Impressions* seem to exist, no significant correlation with *NextDayLeads* exists. The highest such correlation is between *Visits* and *NextDayLeads* at around 0.3. Just like the macro-economic data, the code includes graphs to show these values.

# Data Preparation

The goal for data preparation is to have two final data frames, one for manual and one for website leads. These data frames should only include the dependent variable and all the predictors. It should also not include any empty values.

As mentioned in the EDA, the outliers originating from measurement errors are filtered out. An outlier is defined as lying four times the standard deviation away from the mean. This combination of mean and standard deviation is a common approach (Acuña & Rodriguez, 2004). Typically, the multiplier factor is 2 or 3. Here, a higher value was chosen to not delete to many observations. To take the increasing number of outliers into account, both measures are calculated by year. This way over the whole period 18 observations are filtered out.

All website traffic data prior to the 15th of August 2020 had to be deleted as they display wrong values. Also, for the macro-economic data, the observations from weekends and holidays must be filled out. It was decided to take the values from the previous day, since technically the stock index value, for example, on a Saturday is still the value from Friday. To increase readability, all spaces and special characters from the *Macro* data frame are deleted.

The columns *Impressions*, *Clicks* and *Spend* from the *Ads* data frame were aggregated by day. All values from all funnels and platforms were summed up to create one single observation per day. Afterwards new features based on the *Ads* and *Traffic* and *Leads* data were created. The *rollapply* function was used to calculate averages over the past days. This function works like a rolling window, adjusting the upper and lower bound for every observation. These mean values were calculated for *Impressions*, *Clicks*, *Spend*, *Visits* and *TimeSpent* and *NextDayLeads*. In the last case, a lagging function was necessary to avoid the influence of future values to influence the predictors.

Other features created are *month* and *isNextDayWorkDay*. The first stores the month as a factor and functions as a proxy for season. The latter is a binary variable describing whether the next day is a workday or not, meaning weekend or holiday. This variable was created with a specific *holidays* package. The country code for Germany was specified.

To minimize missing data beyond the joining process, an inner join approach was adopted. This strategy entails retaining only those data rows that are present in all data frames. Through this method, the integrity of the dataset is preserved by excluding rows with incomplete information. Keeping all leads observations would have resulted in almost half of observations having incomplete data. This significant proportion harms the modelling process.

Finally, the data frame has been duplicated, providing separate data frames for both manual and website leads. All empty values, resulting from filtering the outliers, from the dependent variable were deleted. Additionally, the respective other NextDayLeads was eliminated, considering their unavailability in practical scenarios, making predictions impractical. The final data frames comprise 30 columns with 588 and 583 observations for manual and website leads, respectively.

# Modeling

Our model of choice for predicting leads is a random forest since it provides reliable and interpretable results, clear insights into feature importance and benefits from its ability to show complex and nonlinear relations. Keep in mind we are modeling two separate random forests for website and manual lead prediction. However, both models share the same predictor variables and are tuned using the same approaches.

Random forests are made up of a collection of decision trees, in our case regressions trees as the task at hand is regressing the numerical feature *NextDayLeads*. During training, each tree is trained on randomly drawn sub-sample of the data. This method is also called bootstrapping and ensures that the individual trees are not too similar. To further de-correlate the trees, during training each time when a split is performed only a random subset of features (usually a third of all features) is considered to split on. Finally, when predicting values with a random forest, the mean output of all regression trees is computed. Part of the reason why we decided to use a random forest is that they typically offer very good performance without needing complex tuning. However, there are still some hyperparameters that need to be considered when building a random forest.

Before tuning hyperparameters of the actual regression tree, we wanted to find the optimal rolling average window for the dependent variable that is used as an additional predictor. A linear model with a rolling average window of 1-200 days was trained and evaluated on a test set. The left graph in Figure 3 shows the results and suggests that a smaller rolling average window is beneficial. We decided to add a 3-day and a 10-day rolling average to the feature set.

Similarly, the number of trees was determined. By fitting a random forest with 500 trees (with package randomForest) together with test data produces train and test metrics for each added tree. For the manual leads, both test and train RMSE show static behavior, while the test error of around 0.4 is quite high. The test RMSE of the website model however is much smaller and starts to settle around 200 trees and greater at an RMSE of around 0.015. Since random forest typically do not suffer from overfitting, we chose to keep the number of trees at 500 for both models.

After determining the number of trees, other hyperparameters of the random forest models are tuned via grid search. To do so, we define a sequence of possible values for all hyperparameters and create a grid that contains all combinations of those hyperparameter values. The models are then evaluated on each combination. The grid search was applied on the following hyperparameters:

1. The number of variables to randomly sample as candidates at each split. The default value for regression is p/3 = 9 where p is the number of all available variables. For the grid search we tried 6,8,10,12 and 14.
2. The sample size that is drawn for each tree. The default value is 63.2% of the total data. Candidates in the grid search are 10%, 20%, 30%, 40%, 50%,60%, 70% and 80%.
3. The minimum size of terminal nodes. This parameter determines how large the trees grow. A smaller value results in larger trees. The default value for regression is 5. All values between 3 and 20 with 2-step increment were tried.

This time, the model was trained on the entire dataset and the RMSE was calculated on the out-of-bag samples, which means that each data row was only evaluated on trees that do not include it in their random sample. Thus, we can train and test on same data.

To get a better understanding for the optimal parameters we looked at the best ten combinations and chose the models hyperparameters accordingly (see appendix A1&2). For the number of sampled variables 14 is optimal for both the manual and website model. Sample size was optimal at 60% and 50% for the manual and website model respectively. For the node size however a wide range values can be observed in the best ten combinations. Thus, we concluded that the node size is not too relevant for the model’s performance and kept it at its default value of 5.

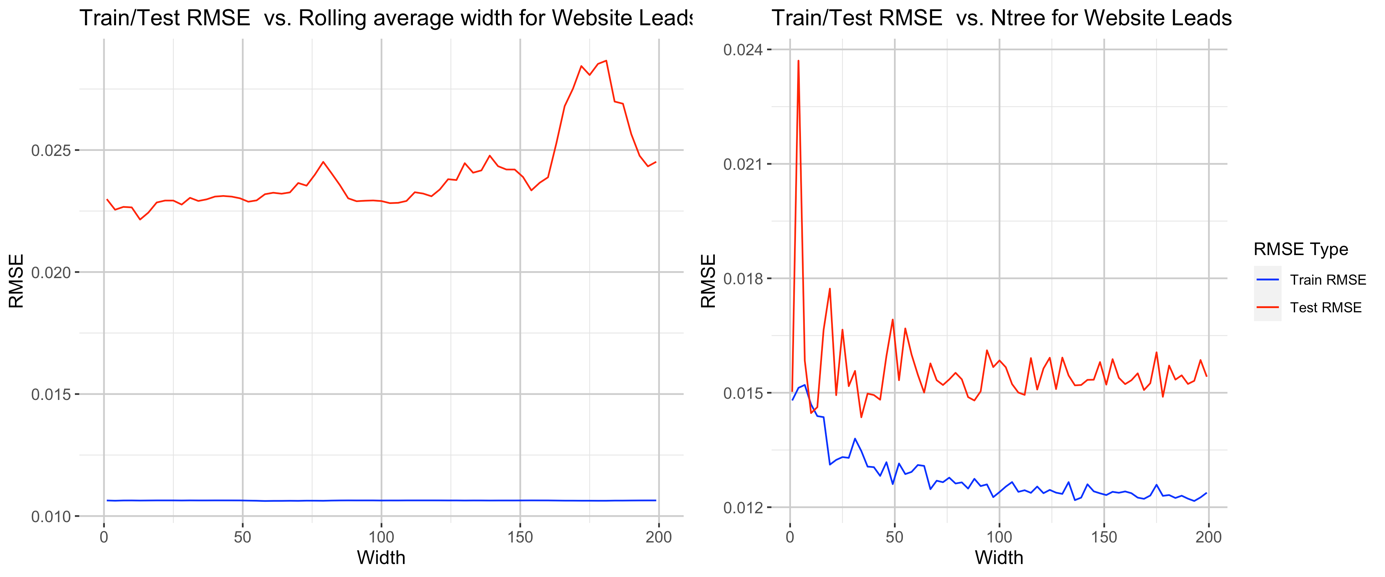


Figure 3: Hyperparameter Tuning of Width Argument and Ntree Setting.

# Evaluation

# Appendix

## A1

Ein Bild, das Text, Screenshot, Zahl enthält.

Automatisch generierte Beschreibung

Appendix A1: Top 10 hyperparameter combinations for the manual leads model.

## A2

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

Appendix A2: Top 10 hyperparameter combinations for the website leads model.