FernUniversität in Hagen

Fakultät für Wirtschaftswissenschaft

Bachelorarbeit zur Erlangung des Grades eines Bachelor of Science

in Wirtschaftsinformatik

über das Thema:

Decision Trees, Random Forest and other Ensemble Learning Techniques

Eingereicht bei:	Univ.-Prof. Dr. Robinson Kruse-Becher
Erstprüfer:	Univ.-Prof. Dr. Robinson Kruse-Becher
Zweitprüfer:	Univ.-Prof. Dr. Andreas Kleine
Betreuer:	M.Sc. Philip Letixerant
von cand.rer.oec.:	Frank Müller
Matr. - Nr.:	9053395
Anschrift:	Calle Castillo 21a 29680 Estepona Spanien
E-Mail:	frank.e.mueller@googlemail.com
Abgabedatum:	02. August 2023
Bearbeitungszeit:	3 Monate
Studienform:	Vollzeitstudium

Table of Contents

List of Figures:

List of Tables:

List of Abbreviations:

colab	Google Colaboratory
LCC	low-cost carrier
LightGBM	light gradient-boosting machine
MCC	Matthews correlation coefficient
ML	machine learning
sklearn	Scikit Learn
XGBoost	eXtreme gradient boosting

1 Introduction

1.1 Motivation

I started the preparation in 2021. One reason was to learn programming by practicing machine learning with Python. One of my projects was the collection of travel data for my own usage.

I live since some years in Southern Spain and study at the FernUniversität Hagen. This requires every semester a journey to Germany, when attending exams in person. The challenge is, how to find the cheapest flight to Germany and back to Spain.

The first impression about the airline prices is, that they are changing seemingly random. But do correlations exist and perhaps we can even predict something in the future. When should we buy the ticket?

During my bachelor study I also attended a course about time series analysis and I was also interested more and more in machine learning, Kaggle competitions and artificial intelligence.

In the last decades in time series forecasting traditional statistical methods like Arima were supplemented by ml approaches. In time series forecasting there is a popular series of competitions called the M competitions. The M4 competition was announced after (Makridakis, et al., 2020) at the beginning of November 2017. The goal was to provide the best forecast for a given data set. “The most important finding […] was that all of the top-performing methods […] were combination of mostly statistical methods…]”. But also, a hybrid method submitted by Slawek Smyl, which used statistical and ML approaches (RNN), had a very good performance. Pure ML methods performed poorly.

This changed in the following M5 competition, which started on March 2020. Under the top 5 there were four teams using LightGBM, a gradient boosting ML algorithm and one team applied a deep learning neural network. (Makridakis, et al., 2022) describes under key findings, that ML methods reign superior this time. Also like in the M4 competition, the combining of models improved the model performance.

1.2 Objectives

Having now an 899 days long dataset of Ryanair flight prices, it is time to explore and analyze the stored data. We use for this the machine learning library “scikit learn” in combination with python, pandas on a Google Colaboratory notebook.

We will try to apply and evaluate classification algorithms based on decision trees and ensemble methods. We will do this under the current best practice of machine learning.

1.3 Problem Statement

Based on the date we will try to predict for future flights of Ryanair to specific locations, if the prices will increase during the next week over a substantial amount. To measure this, we will use standard metrics like the $F_{0.5}$-Score. To be usable, the $F_{0.5}$-Score should be above 0.6.

1.4 Used Tools and Packages

Tool/ Package	Version
Python (ryanair-analysis.ipynb)	3.10.6
Python (ryanair_scraper.py)	3.8
SQLiteStudio	3.2.1
scikit-learn (Pedregosa, 2011)	1.2.2
matplotlib	3.7.1
seaborn	0.12.2
pandas	1.5.3
numpy	1.22.4
pickle	3.11.4
xgboost	1.7.6

Table 1: Used Tools and Packages

2 Pricing and Revenue Management

To understand the use case, let us have a look on the company Ryanair and how pricing in the airline industry work in general.

2.1 Pricing and Pricing Strategies

Ryanair is the most successful LCC in Europe. The short-term goal of a LCC is to fill all seats of the aircraft. The long-term goal can be a high market share, high profit or the fighting against the competitors.

The following chapter is based on and translated from (Meffert, et al., 2008) chapter 4.2. “Preispolitische Entscheidungen”, which focuses on the information relevant to our business case.

In our use case we try to predict price changes. Therefore, it is important to understand the theory behind pricing and revenue management as the underlying pricing strategy.

(Meffert, et al., 2008, p.478) claims that the price determines if product is bought, but also if product is chosen against a selection of products of the competitors and how often the product is bought. The buyers only buy if the value is higher than the costs.

On the pages 484-485 he outlines the factors which influence the price and the change of prices. Relevant for our use case is that our price changes can be reactions to the actions of competitors like price changes or new competitive products. Also, a change of demand to a specific product and change of demand in a market can result in a price change.

The price levels directly influence the demand of the product. The connection between price and demand is expressed in the price elasticity of demand (Meffert, et al., 2008, p.486):

$$price\ elasticity = \frac{percentage\ change\ in\ quantity\ demand}{percentage\ change\ in\ price}$$

The price elasticity is usually negative. When the price falls, then the demand goes up.

The elasticity is influence by some factors, called “Elastizitätsdeterminanten” (Meffert, et al., 2008, pp.490-491):

- Availability of substitution goods: if good can be substituted temporal, local and by products of competitors, then a substitution is possible, and the demand is price elastic.
- Comparable goods also lead to a higher elasticity.
- Is the good durable? If no, the purchase cannot be delayed, the demand becomes price inelastic.
- Is the service or the consumption of the product urgent? With high urgency makes the demand inelastic.
- Marketing of the product has an influence on the price elasticity. If the price is an argument in advertisements, the price elasticity becomes higher, because the customer is more focused on price and reacts to price changes stronger.
- The price of the product also influences the price elasticity, what is shown in Figure 2.

An important element in the pricing strategy is the pricing positioning in the market. (Diller & Herrman, 2003) presents five pricing strategies: the low-price strategy, medium price strategy, the high price strategy, the discount strategy and the cheating strategy (Übervorteilungsstrategie).

LCC are using low price or discount strategies. The low-price strategy provides low quality, low service products for a low price. In a discount strategy a medium quality product is sold for a low price achieved by reduction to the core product properties (no frills), reduced service, efficient processes, and a better cost structure. Ryanair and Aldi are applying this strategy (Meffert, et al., 2008, pp.505-506).

To increase profits Ryanair uses price discrimination (Meffert, et al., 2008, p. 511), it charges different prices for almost identical products. By receiving the consumer surplus Ryanair increases profit. The consumer surplus is the amount of money, what a customer saves because the market price is lower than the price he is prepared to pay.

This only works if individual customers are willing to pay different prices. Therefore, the customers are split into different groups, and charged with different prices.

The types of price discrimination are explained on (Meffert, et al., 2008, pp. 515-517): customers are discriminated by time, by location, by personal properties (age, gender, income, profession), by the number of customers (group price) and by quantity (lower unit price for higher quantities). Yield management applies price discrimination in the service sector.

Often price discrimination require self-selection. The customers are choosing if they accept the price discrimination. Discrimination by time is also common: different prices dependent on the time of purchase. The prices can change during the day, weekdays, seasons, or years and are based on time dependent cost differences (transport and procurement costs for seasonal food) and time dependent preference differences (sale of winter clothing in summer). There are also other forms, which are not relevant to our business case.

In dynamic pricing the prices can also change during the sales period, what is widely used in revenue management.

2.2 Revenue Management

„Yield management is a process by which discount fares are allocated to scheduled flights for the purpose of balancing demand and increasing revenues" (Pfeifer, 1989)

Yield Management and Revenue Management are interchangeable.

(Klein & Steinhardt, 2008, p.2-3) tells the developement of revenue management in the US american airline industry in the 1970s: In 1978 the development of revenue management started with the deregulation of the US air traffic. The airlines were now free to decide about the flight connections and the fares. This resulted in new competitors with lower prices who addressed price sensitive private and business customers with less services and more efficient use of resources.

American Airlines responded with a price discrimination strategy and introduced different rates for different costumer segments. It created a normal tariff for business clients and a cheaper tariff for private clients with restrictions, which made it unattractive for the business customers. The result was a higher capacity utilization, what led to additional tickets sold for marginal cost. American Airline achieved thereby 15% higher sales and passenger volume. The successful execution was copied by other airlines and by companies in the service sector like hotels and car rentals.

(Klein & Steinhardt, 2008, p.8) outline the characteristics of yield management:

- largely fixed capacities
- non-storability of the products and the perishability of capacities after no consumption

- high fixed costs for the capacity provision vs. low marginal costs for the service provision
- strong, simultaneously stochastic, fluctuations in demand
- possibility of pre-booking
- possibility of segment-oriented price differentiation

(Kimms & Klein, 2005, p.9) list the requirements for Yield Management:

1. The creation of the service requires the integration of an external factor, which must be brought in the creation process by the consumer. This can be the consumer by himself or one of his disposal objects.
2. The operational flexibility of the resources, needed for the creation of the service is limited. The capacities cannot be easily adapted to a change of demand.
3. A heterogeneous consumer behavior can be observed when purchasing or consuming the services offered. E.g., the customer has different preferences of the period between purchase and consumption of the service and different individual willingness to pay.
4. The service program must enable the definition of standardized products.

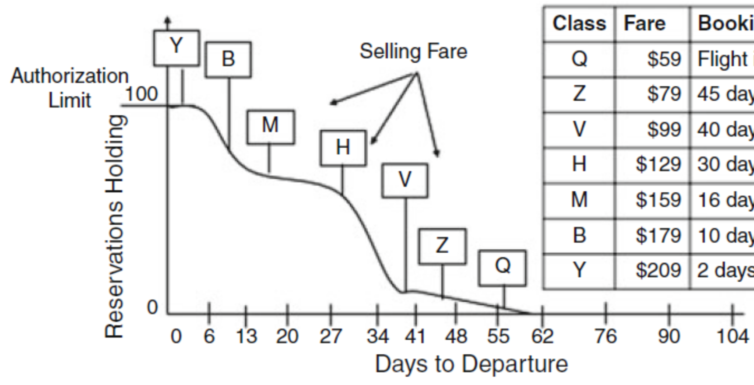
Another important aspect is the danger of cannibalization (Klein & Steinhardt, 2008, p.22), which describes the competing marketing of similar products at different prices. The cheaper product can harm the sales volume of the higher priced product. In the case of LCC the prices close to the departure day are much higher seen in Figure 2. If all tickets are sold before, the cannibalization effect lowers revenue.

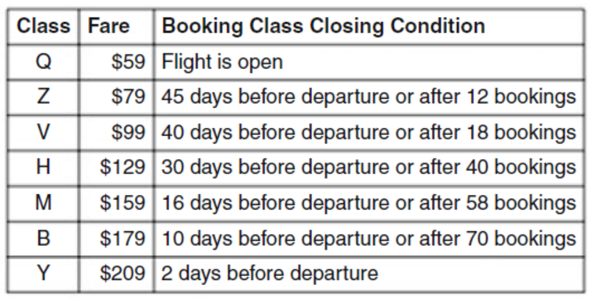
An important long-term goal is a high load factor, which describes the amount of passengers/ number of seats. Figure 5 shows the load factor of Ryanair between 2019 and 2023. To reach a high load factor, flights are usually overbooked by selling more tickets than the flight has capacity. This works most of the time, because on many flights passengers don’t show up (no-shows). But sometimes on oversold flights, if are not enough seats at the time of departure, the airline must apply bumping to reduce the number of passengers.

Revenue management addresses the strategic, tactical and operative level of an airline (Klein & Steinhardt, 2008, p.18-22). The strategic goals are similar to other management strategies: profit maximization, squeezing out of competitors and high long term customer loyalty (CRM). On the tactical/ operative level is the main goal profit maximization. To achieve this every product receives a specific value in the capacity control process. On the one hand, flights have a low variable cost, profit maximization and revenue maximization are correlated in the airline industry. On the other hand, a high load factor can be competing with profit maximization.

By setting the right prices at the right time, airlines try to achieve a high load factor, control capacities and optimize revenue and profit.

In Figure 1 shows an example how to control capacity with different price levels during the sales period. The number of tickets is split into booking classes. If a booking class is sold or a specific date is reached, then the current booking class is closed and a more expensive one is opened.





Class	Fare	Booking Class Closing Condition
Q	$59	Flight is open
Z	$79	45 days before departure or after 12 bookings
V	$99	40 days before departure or after 18 bookings
H	$129	30 days before departure or after 40 bookings
M	$159	16 days before departure or after 58 bookings
B	$179	10 days before departure or after 70 bookings
Y	$209	2 days before departure

Fig. 5.2 Restriction free pricing inventory controls

Figure 1: Capacity Management (Vinod, 2021, Chapter 4)

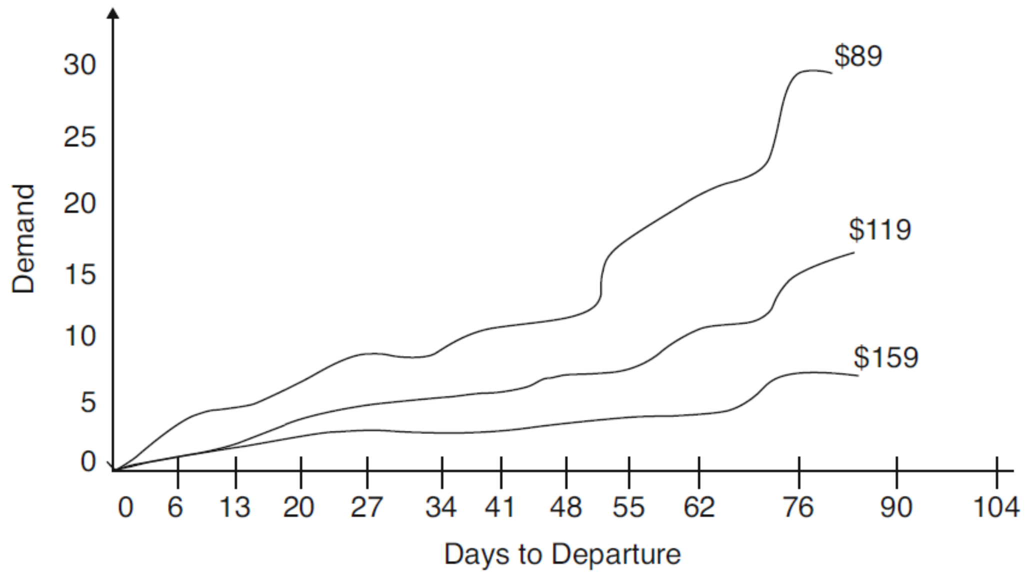


Fig. 5.5 Price sensitivity

Figure 2: Price sensitivity during the sales period (Vinod, 2021, Chapter 4)

The models used by the airlines are much more complex and are using advanced statistical methods to reach the predefined goals.

2.3 Simplified Pricing Model

Because there is only limited amount of information available, namely the prices of past flights and their variation during the sales period, we must simplify the prediction model to a black box model. We don’t have data on the actual demand, the forecasted demand, the cost structure, reactions of competitors and other important model parameters. Therefore, we assume that the system we want to describe is a black box, where we only know as an input the past prices and as an output the current price for a flight.

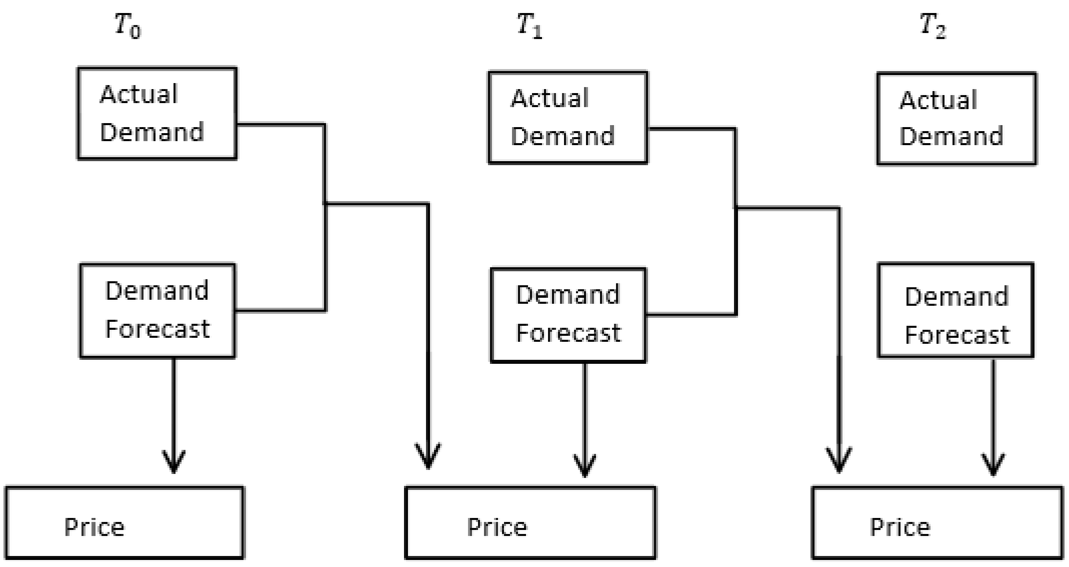


Figure 3: Simplified Pricing Model

We simplify the system in the following way. Ryanair forecasts a demand for a specific period T_0. This demand forecast sets the price for this period. The price of the next period T_1 is now influenced by the demand forecast for T_1, but also by the demand forecast error in period T_0.

When we try to predict price increases, there are two types of influences:

Most of the price changes are planned based on the long-term demand forecast and general planed sales quantities. But when these predictions are wrong, then there will be price changes to balance out the previous prediction errors.

3 Ryanair – Company Profile and Strategy

One important remark before the company presentation: the fiscal year of Ryanair ends in March the same year, so the fiscal year 2022 started in April 2021 and ended in March 2022.

According to (McCarthy, 2022, p.82) Ryanair operates a low fare and offers scheduled-passenger airline serving short-haul, point-to-point routes mainly within Europe.

“Ryanair recorded a loss after taxation of €241m in fiscal year 2022, as compared with a loss of €1,015m in fiscal year 2021. The reduced loss was primarily attributable to a 253% increase in traffic as European Governments eased travel restrictions/lockdowns associated with the Covid-19 pandemic, facilitated by lower fares and offset by strong ancillary revenue performance and cost control within the business.”

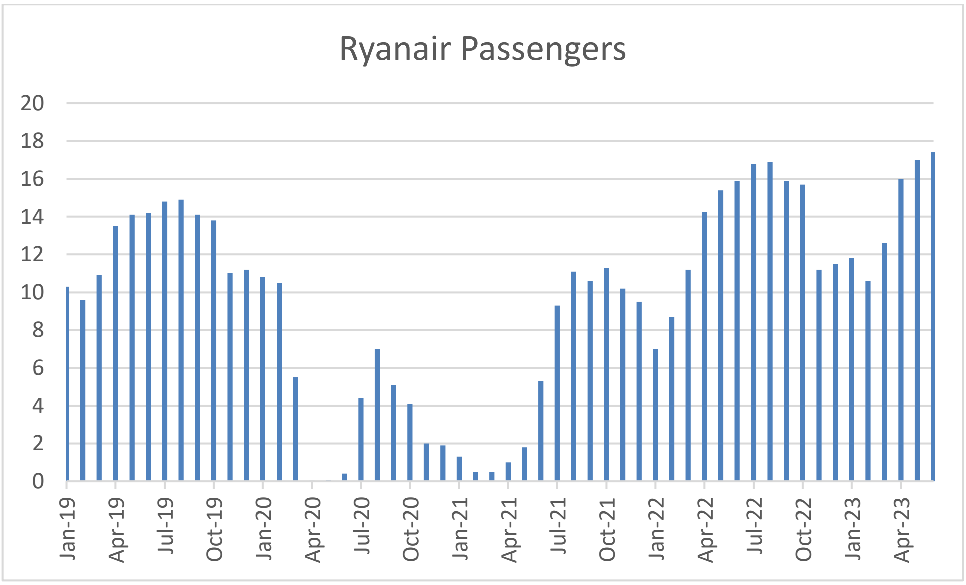


Figure 4: Ryanair passenger numbers between 2019-2023

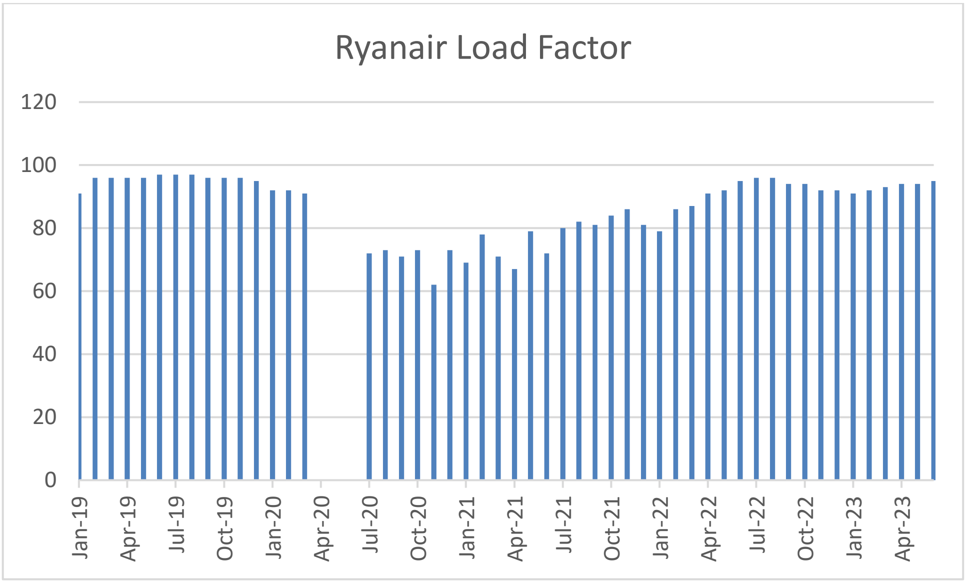


Figure 5: Ryanair Load Factor between 2019 and 2023

In Figure 4 and Figure 5 we can see the performance of Ryanair during the data collection period (January 2021- June 2023). At the beginning of 2020 the Covid pandemic started and influenced Airline traffic in a high amount. The years 2020, 2021 and 2022 were characterized by uncertainty, partial and full lockdowns and an economic downturn. In 2022 the Covid omicron wave hit Europe, which also influenced the Airline industry negatively with flight restriction, Covid tests and a general insecurity about the future. From mid-2021 on, EU Covid Digital Certificates were available, which eased the ability to travel abroad for potential customers. In February 2022 the Russian invasion of Ukraine started, which had also a negative influence on energy prices and overall security.

In Figure 4 we can see that the passenger numbers were almost zero in the first months of 2020 and only recovered the summer season of 2022. We can also see a seasonal pattern, because one of Ryanair most popular connections are the one from northern European cities to Mediterranean vacation spots. While having a high decline of passenger numbers, the airplanes still had a good load factor (amount of passengers/number of available seats in a plane) of over 60%.

Ryanair's general strategy is to establish itself as Europe's largest scheduled passenger airline group.

Selected Key elements (McCarthy, 2022,pp.84-87) of Ryanairs strategy are:

1. Low Fares to stimulate demand.
   "**Ryanair sets fares on the basis of the demand for particular flights and by reference to the period remaining to the date of departure of the flight, with higher fares typically charged on flights with higher levels of demand and for bookings made nearer to the date of departure. Ryanair also periodically runs special promotional fare campaigns."**
2. Ryanair's strategy is to deliver the best *customer service* performance in its peer group.
3. Low operating costs by reducing or controlling the four primary expenses: (i) aircraft equipment and finance costs; (ii) personnel costs; (iii)customer service costs; and (iv) airport access and handling costs.

4 Methodology

4.1 General Methodology

The goal is to create a machine learning model, a mathematical model, which makes predictions based on past data. The models are trained and tested on past data on the training set and evaluated on the test set using predefined metrics.

The training is supervised, the training set consists of input variables called features, predictors or attributes and one output variable called target or label. During the learning process, the predictions can be supervised, because the real values are already known.

In supervised learning there are two main tasks: regression and classification. In time series analysis regression is more common, where the prediction of a target numeric value i.e. prices is the training purpose. For classification tasks the model is trained to predict specific target classes. If there are only two class labels, the process is called binary classification.

In Figure 6 the whole data mining process including model training and model evaluation is pictured.

$$\begin{tikzcd}
\text{Data Collection} \arrow[r] & \text{Data Storage} \arrow[r] & \text{Data Loading} \arrow[dl] \\
 & \text{Preprossesing and Feature Engineering} \arrow[dl] & \\
\text{Hyperparameter Tuning} \arrow[d] & & \\
\text{Training of the different models} \arrow[r] & \text{Model Evaluation} &
\end{tikzcd}$$

Figure 6: Data Mining Process

4.2 Data Collection

The data collection started on 3.1.2021. The first data points after the feature engineering are from 10.1.2021. On 1.5.2023 there were also more flight connections added to check, how good the generalization abilities of the trained models are.

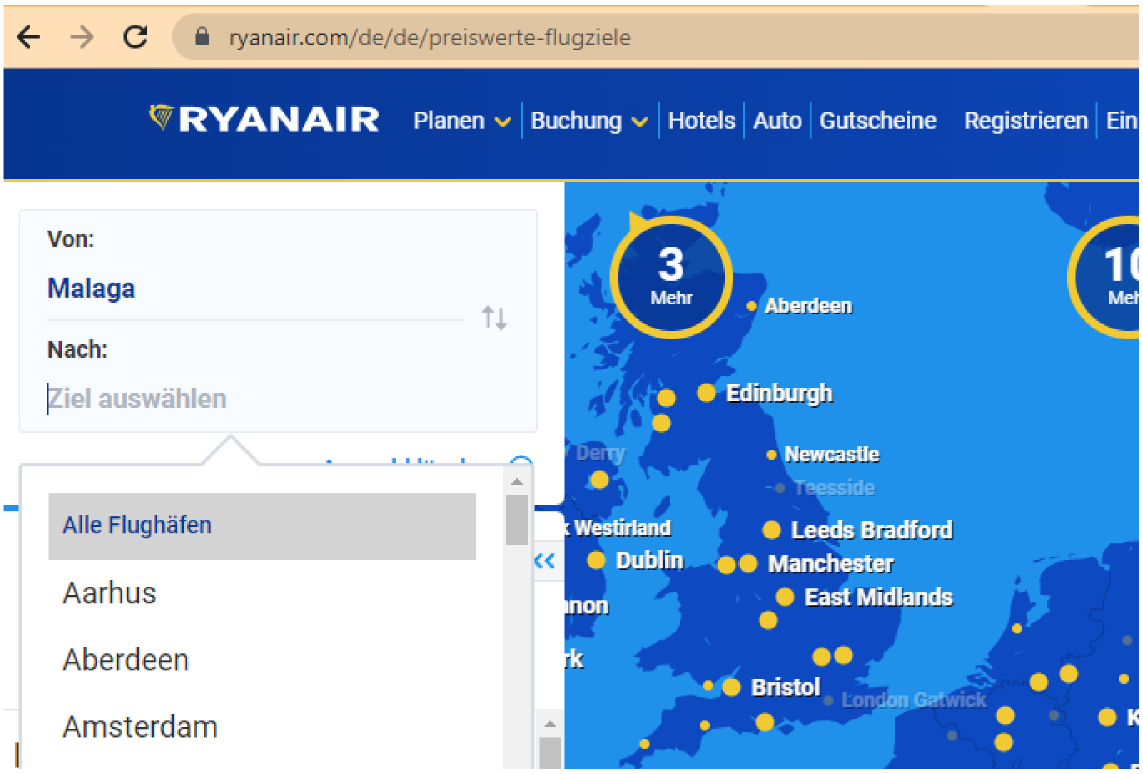
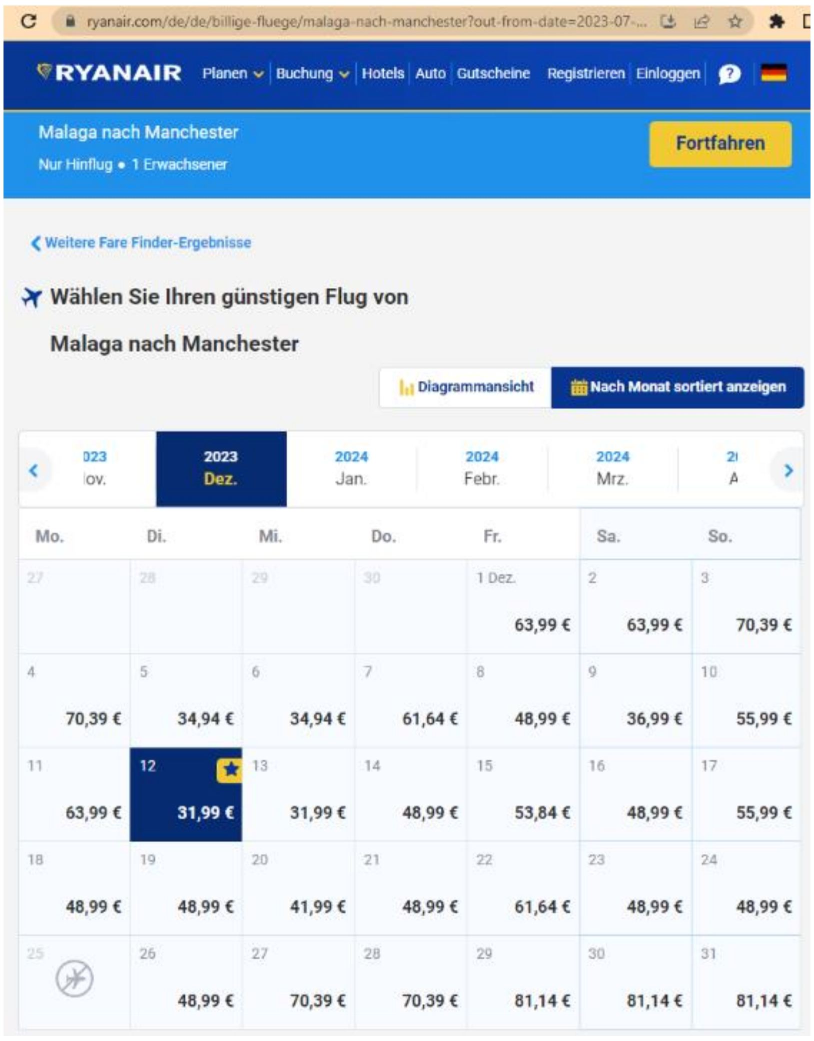


Figure 7: Ryanair Flight Selection Page

The data was collected daily. Some days are missing because of technical difficulties or updates of the used web browser. The script was written and executed in Python (3.8) and Selenium, the data is stored in a SQLite database.

Figure 8: Flight Price Page



ryanair.com/de/de/billige-fluege/malaga-nach-manchester?out-from-date=2023-07-...

RYANAIR Planen Buchung Hotels Auto Gutscheine Registrieren Einloggen

Malaga nach Manchester
Nur Hinflug • 1 Erwachsener
Fortfahren

Weitere Fare Finder-Ergebnisse

Wählen Sie Ihren günstigen Flug von
Malaga nach Manchester

Diagrammansicht | Nach Monat sortiert anzeigen

023 ov. | 2023 Dez. | 2024 Jan. | 2024 Febr. | 2024 Mrz. | 2(A

Mo.	Di.	Mi.	Do.	Fr.	Sa.	So.
27	28	29	30	1 Dez. 63,99 €	2 63,99 €	3 70,39 €
4 70,39 €	5 34,94 €	6 34,94 €	7 61,64 €	8 48,99 €	9 36,99 €	10 55,99 €
11 63,99 €	12 ★ 31,99 €	13 31,99 €	14 48,99 €	15 53,84 €	16 48,99 €	17 55,99 €
18 48,99 €	19 48,99 €	20 41,99 €	21 48,99 €	22 61,64 €	23 48,99 €	24 48,99 €
25	26 48,99 €	27 70,39 €	28 70,39 €	29 81,14 €	30 81,14 €	31 81,14 €

In Figure 8 and Figure 7 the website during the data collection is shown.

4.3 Data Cleaning

During the first month, the website changed the price format. This change is manually corrected. Due to connection and online issues, there were several data points missing.

Due to a configuration error, several connections were captured multiple times. Some days were also scraped multiple times. Both errors led to duplicates, which are also removed. In some of these cases, the price changed during the day. In these cases, the datapoint with the lower price was kept and the others are removed.

Zero values in the price column were caused by connection problems. These instances were also removed. The resulting gaps in the time series of the flights are imputed by interpolation.

In Figure 9 the whole data cleaning process is shown in detail.

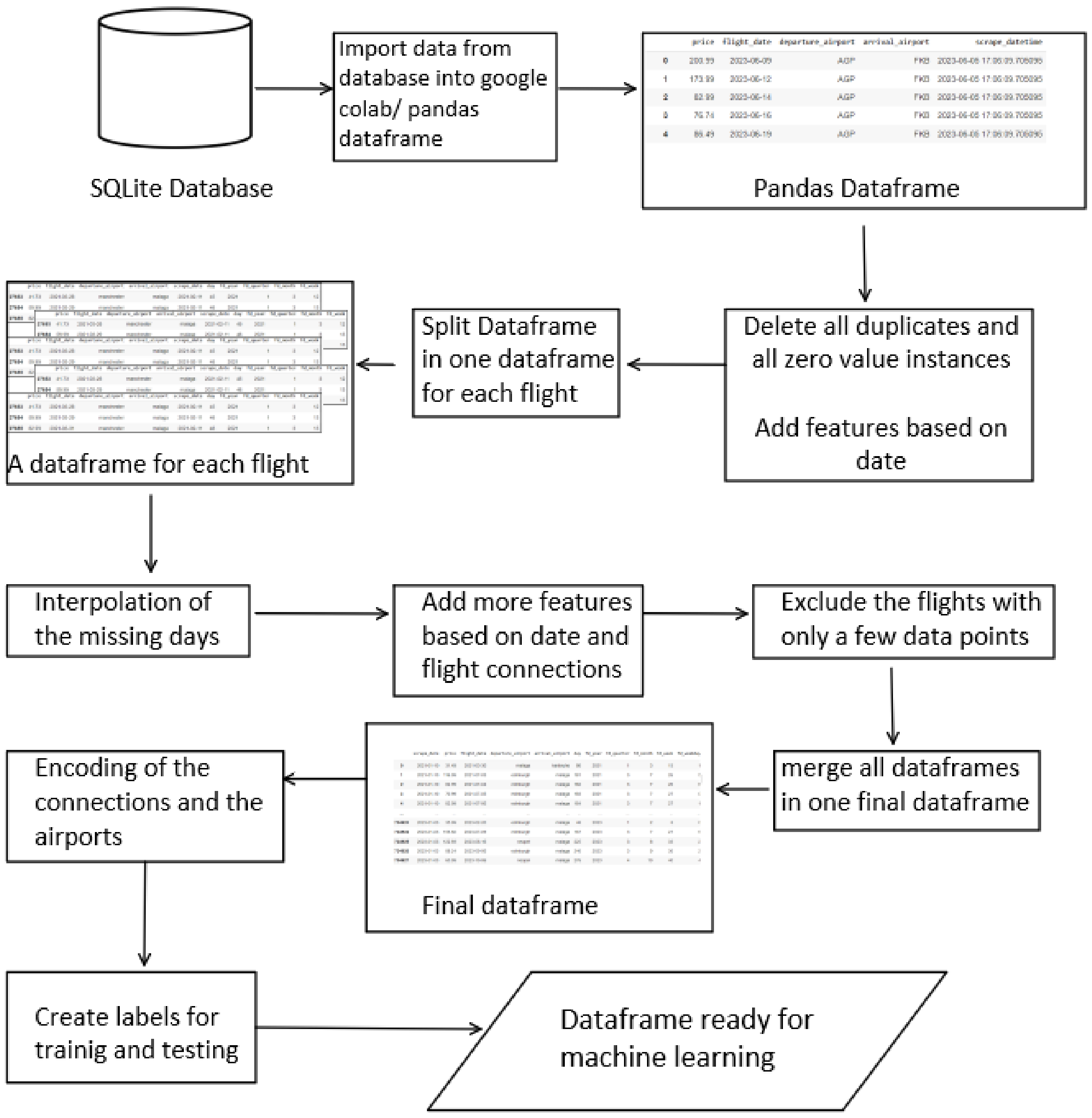

SQLite Database
Import data from database into google colab/ pandas dataframe
Pandas Dataframe
Delete all duplicates and all zero value instances
Add features based on date
Split Dataframe in one dataframe for each flight
A dataframe for each flight
Interpolation of the missing days
Add more features based on date and flight connections
Exclude the flights with only a few data points
merge all dataframes in one final dataframe
Final dataframe
Encoding of the connections and the airports
Create labels for trainig and testing
Dataframe ready for machine learning

Figure 9: Model Building Process

4.4 Feature Engineering

The objective of feature engineering is to "coming up with a good set of features to train on" (Geron, 2022,p.30). He states that this process contains following elements: feature selection, feature extraction and creating new features by gathering new data.

The features scrape_date, price, flight_date, departure_airport and arrival_airport are from the original dataset. The other features are extracted out of these five features.

Here is a brief description of the features of the dataframe:

departure_airport/ arrival_airport:

The departure/arrival airport contains a string of the location of the departure/ arrival airport. Because strings are hard to use in ml-models, they are encoded in binary variables.

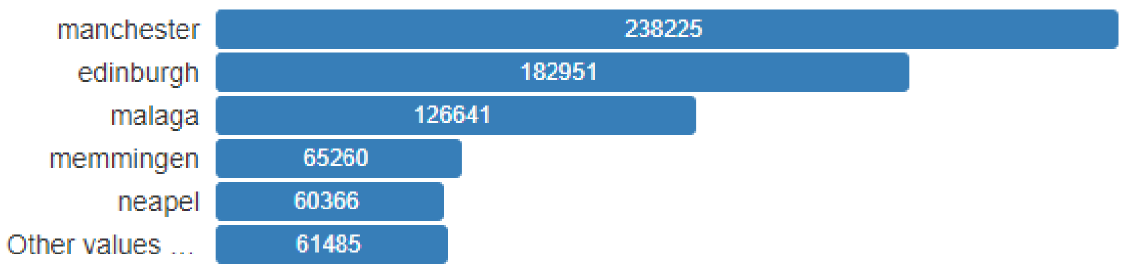


Figure 10: departure airport

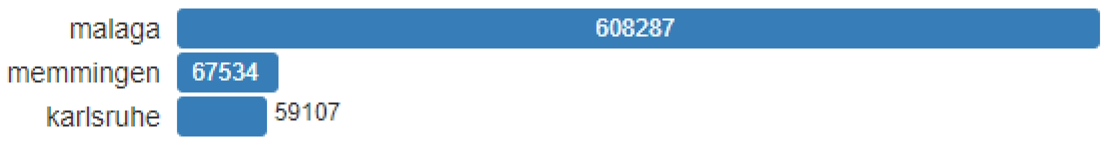


Figure 11: arrival airport

Feature name	**Feature description**	**Format**	**Included in Training/ Test Set**
scrape_date	Date of data collection	datetime64[ns]	no
price	Ticket price of a flight at the scrape_date	float64	yes
flight_date	Date of departure of the flight	datetime64[ns]	no
departure_airport	Departure airport	Object (String)	no
arrival_airport	Arrival airport	Object (String	no
day	Days until departure	int64	yes
fd_year, fd_quater, fd_week, fd_weekday, fd_dayofyear	Year, quarter, week, week-day, day of year of the day of departure	int64	yes
sc_weekday	Weekday of the data collection date	int64	yes
9dayavg	Average price of a specific day of the year	float64	yes
id, connection	Unique Id for every flight/ connection		yes
target	Ticket price 7 days ahead	float64	no
target-perc	Percentage increase of ticket price (to the next week)	float64	no
diff-1, diff-7	Price difference from t0 to t-1/t-7	float64	yes
t-1, t-2, t-3, t-4, t-5, t-6, t-7	Price *- days before	float64	yes
***_departure_airport**	Encoded departure_airport	float64	yes

***_arrival airport**	Encoded arrival airport	float64	yes
1100101, 1100102, …	Encoded connection	float64	no
buy (target)		float64	yes

Table 2: Features of the Dataframe

Buy:

Buy is the target column for classification. It compares the actual price with the future price in seven days. If the future price is at least 15 points and 25% higher, it is labeled as 1 (positive), otherwise as zero (negative). This column reflects the use case.

Day:

Day defines the day till departure or the remaining days of the maximum sales period. The day of departure has the value 0, the previous day 1. Some flights contain over 400 days for sale. Sometimes the sales period stops earlier, probably because all flights were sold or the departure date is later than the last day of the data set.

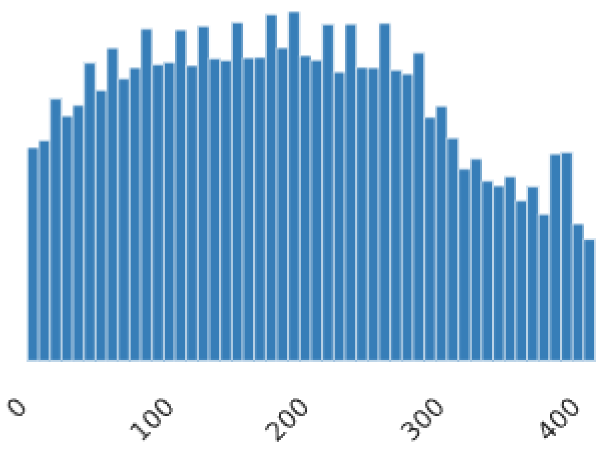


Figure 12: Amount of datapoints for every day till departure

Price:

Price gives the cost of the specific flight on the day of the data collection. For the flights to Warschau, the currency is in zloty, in all other cases in €.

Target:

This column shows the price seven days in the future. It was created as the label for the regression task.

Id/ connection:

Every airport is assigned to a unique identifier (e.g., Malaga hast 100, Karlsruhe has 101). The connection is a number, which consists of “1”+”id of departure airport”+”id of arrival airport”, is for the connection Malaga-Karlsruhe 1100101. The id feature adds the data in the format YYYYMMDD in front of the connection id. For a flight from Malaga to Karlsruhe on 13.7.2022 the id is 20220713100101. These features were added only for faster filtering, but the algorithms used the id column very often.

diff-1, diff-7:

Diff-1 is the difference between the actual price and the price the day before, Diff-7 is the absolute price change over on week.

Lags:

The lags contain price values from previous datapoints.

9dayaverage:

Includes the average sale prices for every day of the year, where the days till departure are between]1;50[and the scrape date is earlier than 4.1.2023. The idea is, that this feature captures the change of prices over the year.

Encoding of airports:

Every airport for arrival and departure has its own feature column. A one means the flight is using the airport, a zero means it doesn’t.

Departure_airport, arrival_airport:

The connections covered till April 2023 for both ways: Karlsruhe-Malaga, Edinburgh-Malaga, Memmingen-Malaga, Manchester-Malaga and one one-way- connection from Neapel to Malaga.

Between 1.5.2023 and 23.6.2023 several connections were added. Both ways: Malaga-Bremen, Bristol- Malaga, Dublin- Malaga, Köln-Manchester, Malta-Malaga and one one-way-connection from Malaga to Edinburgh,

Furthermore, the dates are split into categorical variables:

Out of the dates there are several categorical variables extracted. From the flight date, the year, the quarter of the year, the week, the weekday, day of year are extracted in each own column. There is also on additional column for the weekday of the scrape date.

The final data set is split in three parts: the training set, the test set and another test set 2 with added flight connections.

Due to long training times of the random forest models, several features were cut.

4.5 Exploratory Data Analysis

Table 3: Description of the Training and Test Sets

CRITERIA	TRAINING SET	TEST SET	TEST SET 2
NUMBER OF VARIABLES	43	43	43
NUMBER OF OBSERVATIONS	855036	51086	106890
START DATE	10.01.2021	07.05.2023	07.05.2023
END DATE	30.04.2023	23.06.2023	23.06.2023

The created data frame is split into different sets, used for training and testing.

Table 4: Arrival and Departure Airports in the Training and Test Sets

Airport		Training Set	Test Set	Test- Set 2
Bremen	Departure	0	0	4006
Bremen	Arrival	0	0	3998
Bristol	Departure	0	0	14239
Bristol	Arrival	0	0	14245
Dublin	Departure	0	0	15352
Dublin	Arrival	0	0	15361
Edinburgh	Departure	**205425**	10372	0
Edinburgh	Arrival	0	0	10901
Karlsruhe	Departure	**70507**	4843	4054
Karlsruhe	Arrival	**70377**	4840	4729
Köln	Departure	0	0	10275
Köln	Arrival	0	0	9730
Malaga	Departure	149518	10973	44505
Malaga	Arrival	149518	10973	33597
Malta	Departure	0	0	4729
Malta	Arrival	0	0	4054
Manchester	Departure	**278890**	13996	9730
Manchester	Arrival	0	0	10275
Memmingen	Departure	76843	6137	0
Memmingen	Arrival	79141	6133	0
Neapel	Departure	71624	4765	0
Warschau	Departure	2232	0	0

The test set contains the same flight connections as the training set, but the test set 2 has different characteristics.

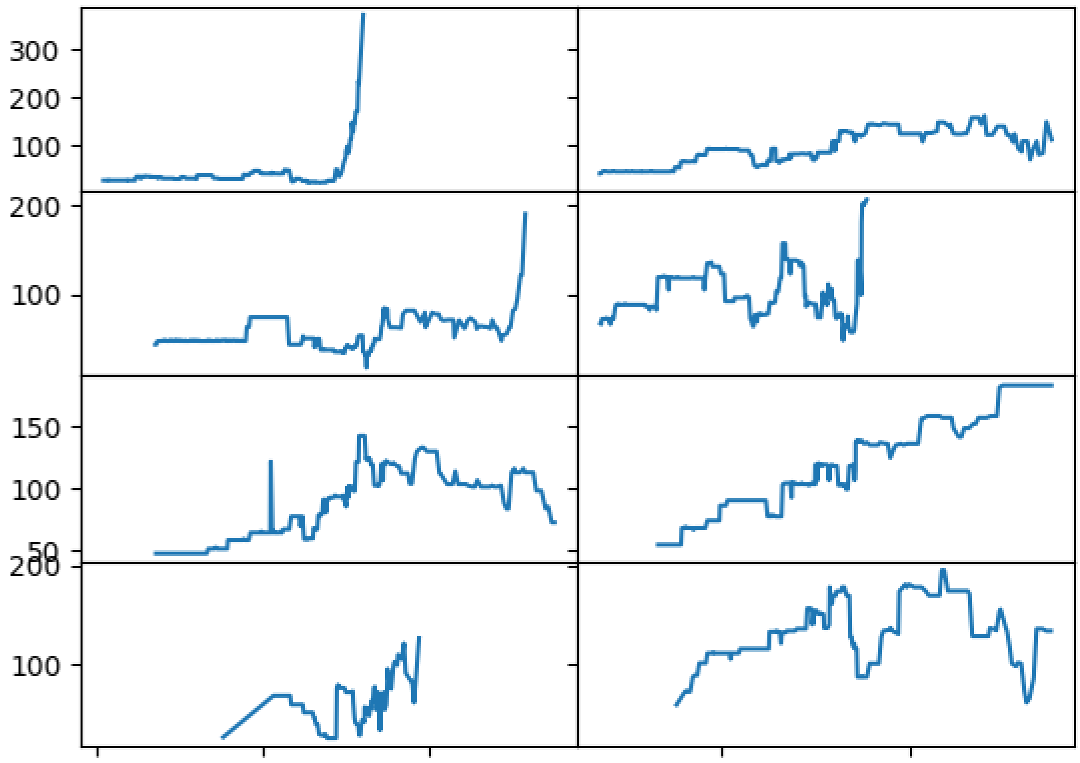


Figure 13: Price development during the sales phase of selected flights

In Figure 13 the progression of flight prices of different random flights is displayed. Often there is a price increase at the end of the sales period. The prices are changing in steps. After the change, there is often a period of a few days without changes. Furthermore, there is a trend visible, in most of the cases the price is increasing during the whole sales period.

Figure 34 shows the development of the prices during the sales period. In Figure 35 the selection of flights is limited on the flights departing in 2022. While it is hard to distinguish specific flights, the general patterns mentioned above are visible.

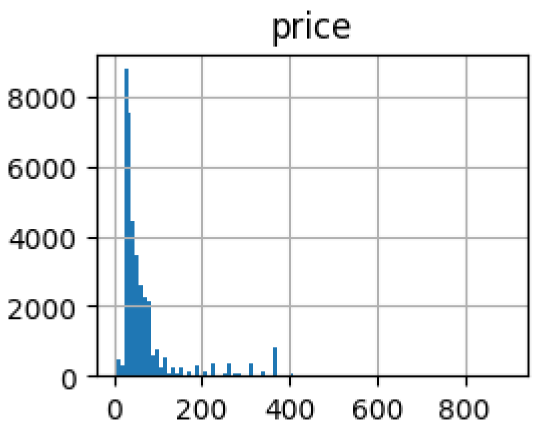


Figure 14: histogram of flight prices

Most of the prices are below 100, what can be seen in Figure 14, with several outliners until 400. The highest value is 901.

The data is very imbalanced, Malaga is overrepresented as arrival airport seen in Figure 11 and Figure 10 shows that most of the flights are starting from Manchester and Edinburgh.

The training set also covers with 2021 and 2022 the covid pandemic, so the training and the test period are not only from different periods, but the periods also itself are probably structural different.

The test set 2 also contains flight connections, which are not part of the training set. This perhaps controls the ability of the model to generalize.

4.6 Hyperparameter Optimization with Grid Search on the Training Set

The data is split into a training set and a test set. To find the best hyperparameters for each model, the training set is again split into subsets (containing each a training a validation set, Figure 15), which are used by the sklearn function GridSearchCV for time series cross validation. Between each training set and each validation set there is a time gap of seven days. The different test sets cover also in different time periods. The models are trained on all training sets and evaluated on the corresponding validation set. The result is the average of the validation results. In our case there are three sets. There is also a time gap between the latest validation set and the final test set.

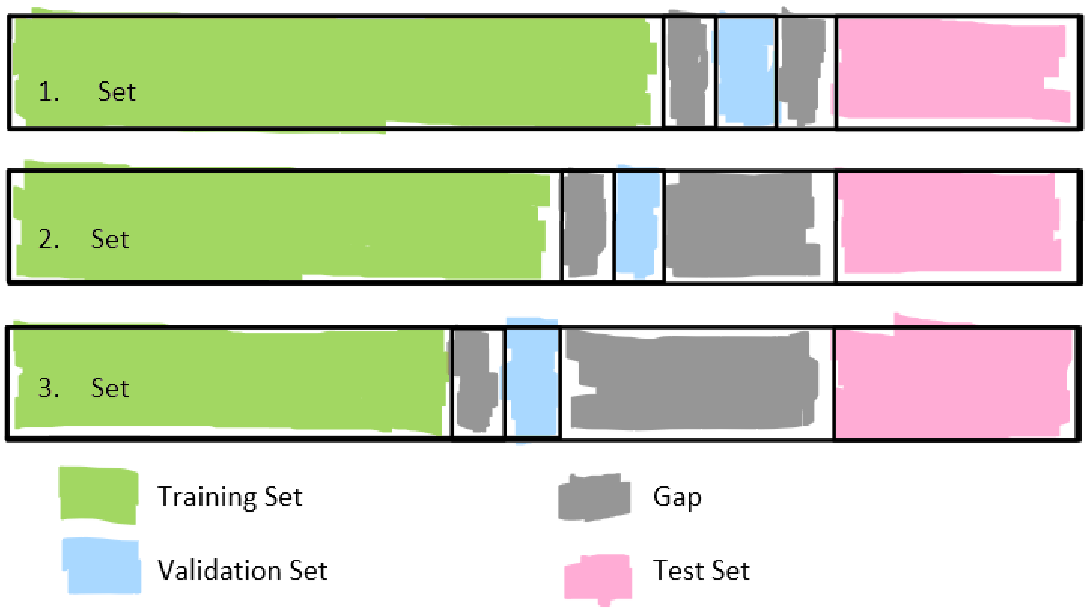


Figure 15: The different training/ validation sets used for GridSearchCV

GridSearchCV takes parameter value combinations as an input and gives a predefined metric as an output for each parameter combination. From these results the next parameter combinations are chosen manually to find out the optimal model hyperparameters.

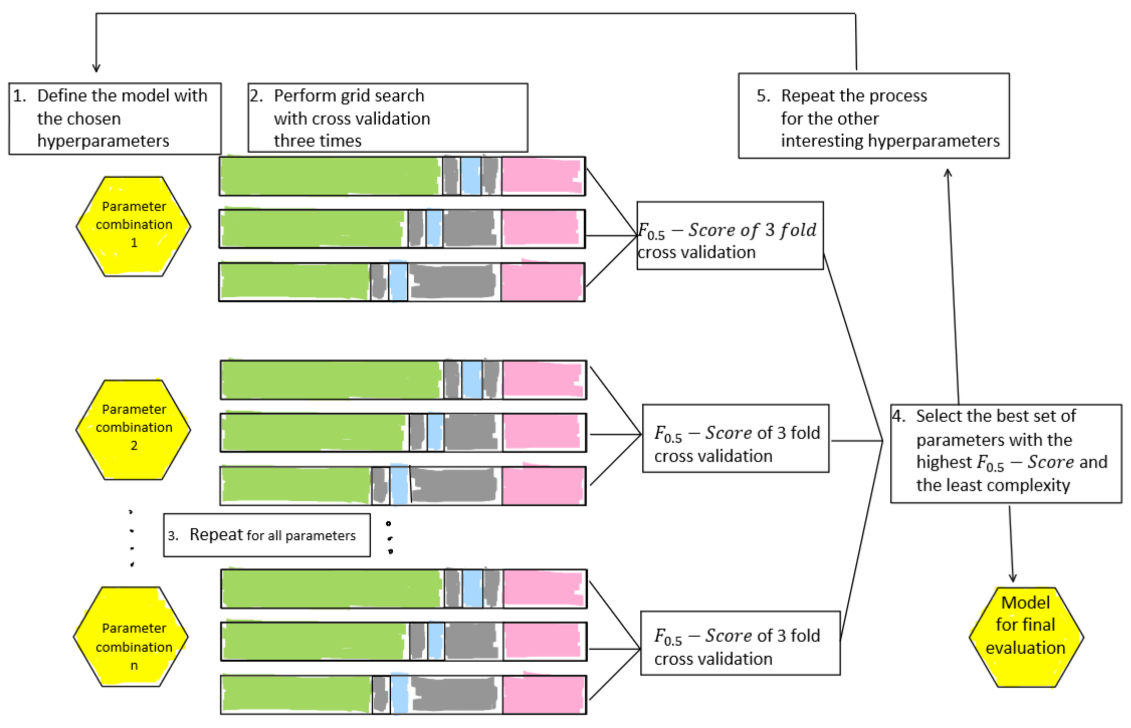

1. Define the model with the chosen hyperparameters
2. Perform grid search with cross validation three times
5. Repeat the process for the other interesting hyperparameters
Parameter combination 1
Parameter combination 2
Parameter combination n
$F_{0.5} - Score\ of\ 3\ fold$ cross validation
$F_{0.5} - Score$ of 3 fold cross validation
$F_{0.5} - Score$ of 3 fold cross validation
3. Repeat for all parameters
4. Select the best set of parameters with the highest $F_{0.5} - Score$ and the least complexity
Model for final evaluation

Figure 16: Hyperparameter optimization

The metric we are using is the $F_{0.5}$-score.

When we chose the hyperparameters manually, we only could try one or two, because Google Colab stopped the calculation sometimes, when it was too long. The found parameters are probably good but not optimal.

The trained algorithms have often many hyperparameters from which not all were chosen. From the ones not tested, the standard values were used.

To choose the best hyperparameter, a high $F_{0.5}$-score and a low model complexity was preferred. The process can be seen in Figure 16.

4.7 Training on full Training Set

After the best hyperparameters for each algorithm were found, a final model for each algorithm was trained on the full training set, which is also separated by a gap from the test set to avoid data leakage as shown in Figure 17.

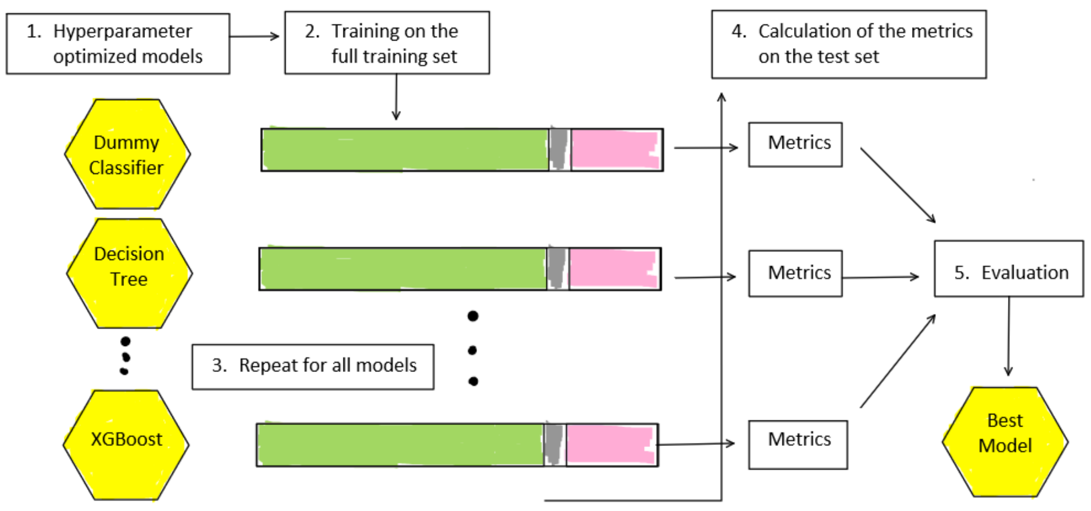

1. Hyperparameter optimized models
2. Training on the full training set
4. Calculation of the metrics on the test set
Dummy Classifier
Decision Tree
XGBoost
3. Repeat for all models
Metrics
Metrics
Metrics
5. Evaluation
Best Model

Figure 17: Model Training and Evaluation

4.8 Model Assessment and Model Selection on the Test Set

The trained models now are evaluated on the training set. The main evaluation metric is the $F_{0.5}$-score, but also precision, recall, F1-Score, Matthews Correlation, $F_{0.5}$-Score on the test set are also calculated for further analysis.

To check how good the generalization of the model works, the same process is repeated on the test set 2, which consists of flight connections not included in the training data.

4.9 Threshold optimization on the Test Set

For further optimization, the idea is to change the decision threshold from 0,5 to a better one. A change in the threshold changes the recall and the precision of a model on the evaluation set and thus also the $F_{0.5}$-score.

To find out the optimal threshold, we calculate for every threshold the corresponding $F_{0.5}$-Score on the training set. The threshold with the highest $F_{0.5}$-Score is now selected. Then we use this new optimal score on the test set to evaluate, if the $F_{0.5}$-Score is increased compared with the previous case with the standard threshold of 0,5.

5 Performance Measures

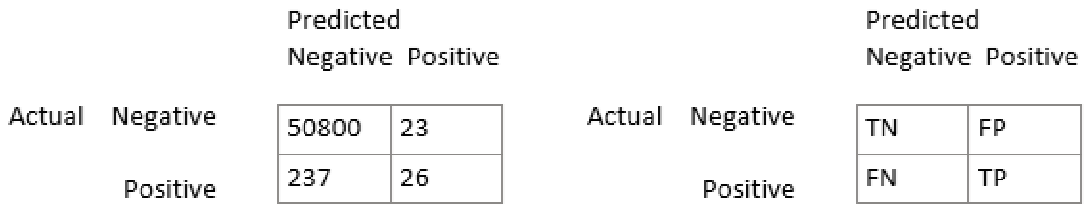
The task is a classification task and to evaluate the models for hyperparameter optimization and in the final evaluation, we need to choose metrics dependent on the use case.

We must take into consideration that we have imbalanced data (8761 true observations against 1009079 false observations in the training set) and two classes.

Because we have a binary classification problem, a good start to approach this problem is a two classes confusion matrix for a base evaluation.

5.1 Confusion Matrix and associated Metrics

To create a confusion matrix, the predictor predicts the classes and then the predictions are compared to the actual observations. Each row stands for an actual class and each column represents a predicted class.



		Predicted Negative	Predicted Positive
Actual	Negative	50800	23
	Positive	237	26

		Predicted Negative	Predicted Positive
Actual	Negative	TN	FP
	Positive	FN	TP

Figure 18: Example for a Confusion Matrix

There are two possibilities for class forecasting: a direct approach or with a threshold. In the direct approach the model predicts either 0 (negative) or 1 (positive). When working with a threshold, the model predicts a probability between [0,1]. Then a decision threshold is defined between [0,1]. If the prediction is now lower than the threshold, it predicts a 0, if it is higher, it predicts a 1. When using a threshold of 0,5, the results are the same like with the direct approach.

In our binary classification 2x2 matrix we have four different cases: TP and TN are used, where the predictions match the observations. FP is a wrong classified negative observation and FN is a false classified positive observation. The following metrics are all based on the confusion matrix.

5.1.1 Why not use Accuracy?

The first and obvious metrics is called accuracy. It describes the percentage of the correct predictions to the total amount of predictions.

$$accuracy = \frac{TN + TP}{TP + TN + FP + FN}$$

But because we have an imbalanced dataset, prediction always false leads to a very high accuracy. But the use case is to predict if we should buy a flight. So, the interest is to predict the positive observations and the negative observations are of a minor interest.

5.1.2 Recall and Precision

The recall and the precision are performance metrics focused on the correct predicted positive observations. The recall (also sensitivity, true positive rate) describes the ratio of positive instances that are correctly predicted by the classifier. In this case it gives an answer to the question: “How many of the relevant price increases catches our predictor?”.

$$recall = \frac{TP}{TP + FN}$$

The precision is focused on the accuracy of the positive predictions. It shows the ratio of all correct classified positive observations to all positive observations. It answers the question: “If the prediction is true, how likely is it to be correct?”

$$precision = \frac{TP}{TP + FP}$$

For our use case recall and precision are both important, while precision matches the use case much better and should be weighted higher in the final metric.

5.1.3 Recall/ Precision Trade Off and the Precision-Recall Curve

The threshold, precision and recall are all connected with each other. If the predicted probability is greater than the threshold, the predicted class is positive, otherwise it is negative. By decreasing the threshold, more instances are classified and misclassified as positive. This also means, that the more the threshold is decreased, the precision increases and the recall decreases. The precision-recall curve displays for every threshold the recall/precision combination with a datapoint. The datapoints of the neighboring thresholds are connect.

Figure 22 is an example of a bad performing curve. The closer the graph is to the abscissa and the ordinate, the worse it is performing.

For every threshold, also a F-score can be calculated. Finding the maximal F-score can find the optimal threshold for an optimal precision recall combination.

5.1.4 F- Score

The F-Score gives a balanced metric, which includes both the precision and the recall. If one of both is low, the F-Score also becomes low. The introduction of β takes gives now recall and precision a different weight in the F-Score.

$$F_\beta = (1 + \beta^2)\frac{precision \times recall}{\beta^2 precision + recall}$$

With a $\beta = 1$, precision and recall are considered equally important.

$$F_1 = (1 + 1^2)\frac{precision \times recall}{1^2 precision + recall} = 2\frac{precision \times recall}{precision + recall}$$

In our use case precision is much more important, so the weight of the recall is reduced by setting β to 0.5.

$$F_{0.5} = 1.25\frac{precision \times recall}{0.25 \times precision + recall}$$

5.2 Threshold optimization with the Precision- Recall Curve

Classification models in binary classification problems usually are using a decision threshold of 0.5. By shifting the threshold, the precision and recall can be changed, what results in a different F-Score. Because the relevant metric is the $F_{0.5}$-Score, the last step of the training process is the search for the optimal decision threshold of each model.

In the implementations all precision, recall and $F_{0.5}$-Scores are calculated for every threshold in the training set. Then the threshold of the highest $F_{0.5}$-Score is selected. This threshold should in the evaluation phase improve performance on the test set of the trained model further.

5.3 Matthews Correlation Coefficient

(Matthews, 1975) tried to find an accuracy metric, which applied on imbalanced class case like this one. While the previous metrics use TP, FP and FN, the MCC also uses TN for the calculation. It ranges from [-1,1]. -1 is an inverse correlation, 0 a random prediction and 1 shows a perfect correlation.

$$MCC = \frac{TP \ \times TN - FP \ \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The MCC gives a balanced overall view about the quality of the predictions.

6 General concept of Ensemble Learning

6.1 Overfitting, Bias and Variance – the Ideas behind Ensemble Learning

One goal in machine learning is to create a model which can make predictions with previously unknown input data. Therefore, observations are split into a training and a test set. The model is trained on training set. The error on the training set is called training error. This error decreases with the complexity of the model. In case of decision trees, the training error can be zero if the tree is deep enough and every instance has its own leaf.

If the model is tested now on the test set, the error of the predictions is called the test error. With increasing model complexity, the training error usually decreases, while the test error increases. This effect is called overfitting, which describes, that the model’s ability to generalize is weak.

To avoid overfitting (Geron, 2022,p.31) proposes to fight overfitting:

- by simplifing “the model by selecting one with fewer parameters, by reducing the number of attributes in the training data, or by contraining the model.”
- by gathering more training data and
- by reducing the noise of the training data.

The reason of the split between training and test set is to avoid overfitting. Overfitting places a big role in the hyperparameter optimization phase.

Another aspect in machine learning is the difference between bias and variance. If training sets are from the same population, high variance models have highly different predictions. Variance provides an estimate of how much the estimate varies as we vary the training data. A high variance also correlates to overfitting.

High biased models fit very good to the training set. On different training sets on the same population, they produce very different models. Bias is the difference between the average estimator from different training samples and the true value. High biased models tend to underfit.

(James, et al., 2013, p.33) explains how the squared error of a regression model can be decomposed into a bias part and a variance part, and how variance and bias relate to each other:

In case of a regression problem, he assumes that $Y = f(X) + \epsilon$, where $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$. So, he calculates the expected prediction error of a regression fit $\hat{f}(X)$ at an imput point X=x_0, using squared- error loss:

$$E(x_0) = E[\left(Y - \hat{f}(x_0)\right)^2 |X = x_0]$$

$$= \sigma_\epsilon^2 [E\hat{f}(x_0) - f(x_0)]^2 + \mathrm{E}[\hat{f}(x_0) - E\hat{f}(x_0)]^2$$

$$= \sigma_\epsilon^2 + \mathrm{Bias}^2\left(\hat{f}(x_0)\right) + \mathrm{Var}\left(\hat{f}(x_0)\right)$$

$$= \mathrm{Irreducible\ Error} + \mathrm{Bias}^2 + \mathrm{Variance}$$

This also applies for classification cases. The error is also split into three parts: the noise, bias and variance.

To minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias.

(James, et al., 2013, p.34) defines variance and bias the following:

Variance is the amount by which $\hat{f}(x_0)$ would change, if we estimated it using a different training data set. For high variance models applies, that small changes in training data can result in large changes in $\hat{f}(x_0)$.

Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.

(James, et al., 2013, p.35) claims that the more flexible methods become, the more the variance increases and the bias decreases. The overall goal is a good test set performance with low variance and low bias. But in reality, models have either a low variance or a low bias on the test set. This is effect is called the bias-variance trade-off.

So how can we reduce bias and variance? In the following chapters some of the methods are introduced: creating ensembles, bagging, boosting, the random subspace method and stacking.

6.2 Ensemble and Voting Classifiers

(Geron, 2022, p.211) defines an ensemble as a group of predictors. The final prediction is extracted on different ways out of the predictions of the single predictors. The idea is to create a strong learner out of a group of weak learners.

The most obvious one is by voting. If the final prediction is determined by majority vote of the predictions of the ensemble predictors, the classifier is called a hard voting classifier. If all classifiers of the ensemble also offer a probability prediction, these probabilities can be averaged as a result. This technique is called soft voting.

In the ensemble there can be classifiers of the same kind or of different kinds. We are using in the soft voting and the hard voting model the prediction results of the trained decision tree, random forest, AdaBoost and XGBoost models.

The random forest, AdaBoost and XGBoost model are also ensembles by themselves. AdaBoost and XGBoost use additionally weighted combination of predictors.

6.3 Bagging and Pasting

(Geron, 2022, pp. 215) argues, that if we train different models on different subsets our data, we create different classifiers. Sampling with replacement is called bagging (**b**ootstrap **agg**regation), without replacement it is called pasting. Training models on the different subsets creates different predictors. By aggregating the predictions of the predictors, the “aggregation reduces both bias and variance” compared to the individual predictors.

The problem of decision trees is the high variance, so when we train two models on a split dataset, we will receive different results.

(Hastie, et al., 2009, pp. 282-283) explains this on a regression exemple: when he fits to the training data $Z = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, he receives $\hat{f}(x)$ for an input of x. He creates different subsets $Z^{*b}, b = 1,2, \dots, B$, with bagging of the data. For each subset he creates a new model with the prediction $\hat{f}^{*b}(x)$. The bagging estimate is the average of all predictions:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

In a classification scenario, we count the classes predicted by each model and then take a majority vote of the ensemble to choose the selected class.

Bagging leads after (Hastie, et al., 2009, pp. 587-588) to a reduction of variance. A deep grown tree has usually a low bias with a high variance. By bagging, the bias of the bagged trees is kept low, but the variance is reduced. Lets assume we have B trees each with a variance of σ^2, the average variance is $\frac{1}{B}\sigma^2$. But because the trees are positivly correlated with ρ, the average variance is:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

With a high number of trees (B) only the first term stays significant. So increasing B decreases the variance.

Bagging is used in Random Forests.

6.4 Random Subspace Method

The second idea to create Random Forests is the Random Subspace Method developed by (Ho, 1998). He proposes to train trees on the whole data, but by randomly selecting of a small number of features he creates different subsets, on which decision trees are constructed. This results in a large variety of different trees. He calls the method “decision forest”.

To use this method in the sklearn random forest algorithm, max_features must be “sqrt” or “log2” to lower the feature space.

6.5 Boosting

Boosting is not like bagging with bootstrap samples. It uses weak predictors to combine them into one stronger predictor. According to (Geron, 2022,p.222) the predictors are trained sequentially, and each predictor tries to correct its predecessor. So, the later predictors contain information of previous predictors and their errors. The initially dataset is modified for each tree.

We implemented two popular algorithms AdaBoost and XGBoost (based on gradient boosting). AdaBoost pays attention on training instances, where its predecessor underfitted. In gradient boosting the focus is on the residual error’s previous predictor.

6.6 Stacking

An extension to voting is stacking (stacked generalization). While voting aggregates the predictions, (Geron, 2022, p.232) explains that is perhaps better to train first different models and then train a model, which uses the previous predictions as input for final prediction.

The final predictor is called a blender.

7 Ensemble Learning Algorithms

In the code section there are regression and classification models. Because of the bad performance of the regression models, the use case is a classification model. The algorithms work both on regression and classification tasks and are first explained on a regression case and later enlarged on a classification case.

7.1 Decision Trees with CART

While a decision tree is not an ensemble learning algorithm, many ensemble algorithms are based on decision trees. They can also be used with other algorithms, but decision trees are more common.

A decision tree consists of nodes. The tree has a root node on the top. Connected to this node there is a special test on a feature, which leads to additional nodes. This process is repeated, until it reaches the last node, the leaf node, which contains the final prediction or the prediction probability. The leaf node has no test attached.

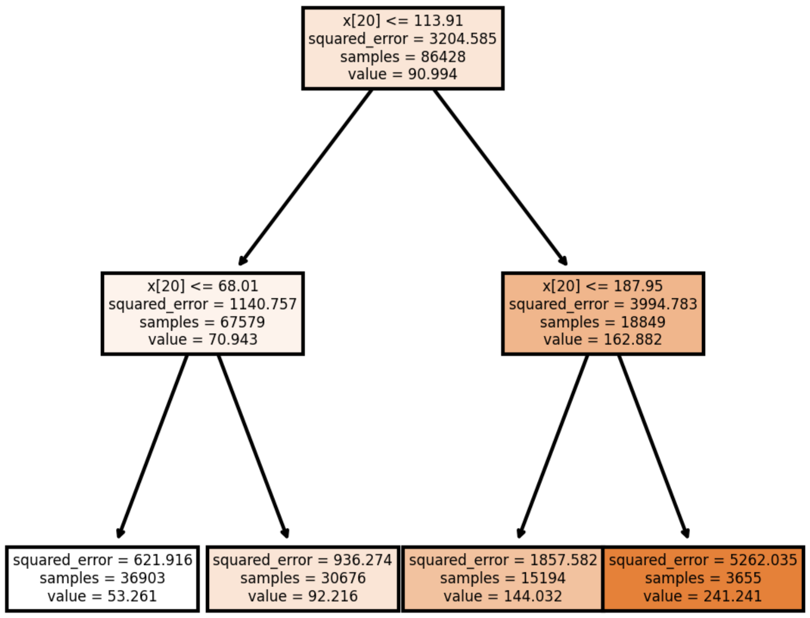


Figure 19: Example of a Regression Tree

We suppose that a new instance has the feature x[20]= 200 and we try to use the decision tree in Figure 19 to make a prediction.

In the regression tree in Figure 19, we can see the first split on feature x[20] at 113,91. Because the instance is greater than 113,91, the next node is on the right branch. There we can see that the split is at 187,95. So here we are also using the right node and we arrive at the leaf node. There the predicted value is 241,241, which is the average value of the 3655 training samples.

To create the optimal decision tree, a brute force approach can be applied. Greedy algorithms like CART, ID3 and C4.5 are much faster and are commonly used.

To grow a regression tree with the CART algorithm (Classification And Regression Tree) developed by (Breiman, et al., 1984), described in (Hastie, et al., 2009, p.307) in chapter 9.2 “Tree-Based Methods”. He uses data with N observations, which consists of p inputs and a response: $(x_i, y_i) for\ i = 1,2, \dots, N\ with\ x_i = (x_{i1}, x_{i2}, \dots, x_{ip})$. The algorithm now must build the decision tree by deciding, which variable where to split.

“Suppose first that we have a partition into M regions $R_1, R_2, \dots, R_M$ and we model the response as a constant in each region:”

$$f(X) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

In this case it is a regression task, so the minimization criterion is the sum of squares $\sum(y_i - f(x_i))^2$. The best $\hat{c}_m$ is the average of y_i in region R_m:

$$\hat{c}_m = ave(y_i | \, x_i \in \, R_m).$$

He now searches for a splitting variable j and a splitting point s, which are defining a pair of half-planes:

$$R_1(j,s) = \{X | RX_j \le s\} \; and \; R_2(j,s) = \{X | RX_j > s\}$$

Now he searches the splitting variable j and the split point s, that are solving the minimization problem:

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_{2(j,s)}} (y_i - c_2)^2]$$

This is solved by $\hat{c}_1 = ave(\, y_i | \, x_i \in \, R_1(j,s)) \; and \; \hat{c}_2 = ave(\, y_i | \, x_i \in \, R_2(j,s))$

After he found the best split, this is repeated on all the resulting regions until the breaking criterion like maximal depth is reached.

In a classification setting like in the use case of this thesis, we use a different impurity measure (Hastie, et al., 2009, p.309). In chapter 9.2.3 “Classification Trees” the proportion of class k observations in node m is:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

The majority class is classified by:

$$k(m) = \arg max_k \hat{p}_{mk}$$

Now there are different measures $Q_m(T)$ to display the impurity of a node. The most used is the Gini index, that is also applied in our models:

$$Gini\ Index: \sum\nolimits_{k \neq k\prime} \hat{p}_{mk} \hat{p}_{mk\prime} = \sum\nolimits_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Another way to interpret the Gini index is not to classify the observations to the majority class in the node, but to classify them to class k with probability $\hat{p}_{mk}$. Then we can define a threshold, and if $\hat{p}_{mk}$ is over the threshold, the whole node is classified to class k.

(James, et al., 2013) describes in chapter 8.1.4 the advantages and disadvantages of trees:

Trees are easy to explain and easy to display graphically. So, they can be interpreted without a great mathematical knowledge. The tree can also handle qualitative predictors without creating dummy variables and the tree structure can even deal with missing values. Because the tree basically only contains if-conditions, the calculation time is very fast.

But because a decision tree is a simple method, they often underperform against other more complex and specialized methods in predictive performance.

(Hastie, et al., 2009) argues in chapter 9.2.4, that decision trees are also very non robust, so a small change of input can lead to a completely different leaf, which has a different output, which leads to a low predictive accuracy and a high variance. If the tree becomes too complex, it overfits to the data and is not able to generalize well. The trees also have problems to capture additive structures.

To address these problems, more advanced algorithms are developed based bagging, boosting and stacking.

The predicted class probability is the fraction of samples of the same class in a leaf.

7.1.1 Implementation

Hyperparameters (default value)	**Value**
Split criterion (gini)	gini
Splitter (best)	best
max_depth (None)	8
min_samples_split (2)	2
min_samples_leaf (1)	60
min_weight_fraction_leaf (0,0)	0.0
max_features (None)	None
random_state (None)	None
max_leaf_nodes (None)	None
min_impurity_decrease (0,0)	0,0
class_weight (None)	None
ccp_alpha (0.0)	0.0

Table 5: Hyperparameter Decision Tree

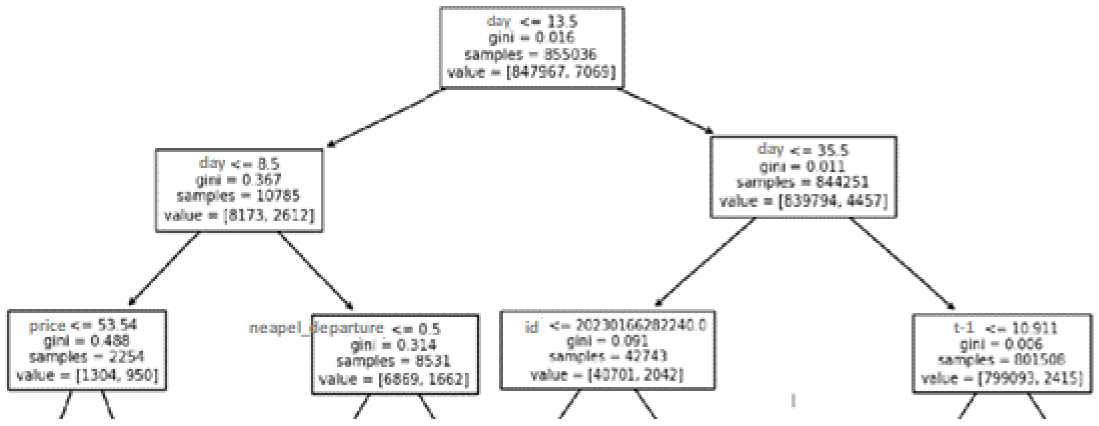


Figure 20: First three levels of the Classification Tree Model

In Figure 20 and Figure 21 the feature "day" is dominating the decision tree. It splits the data in the first two levels. "Price" also has some importance. The domination of one feature can cause problems with the following algorithms.

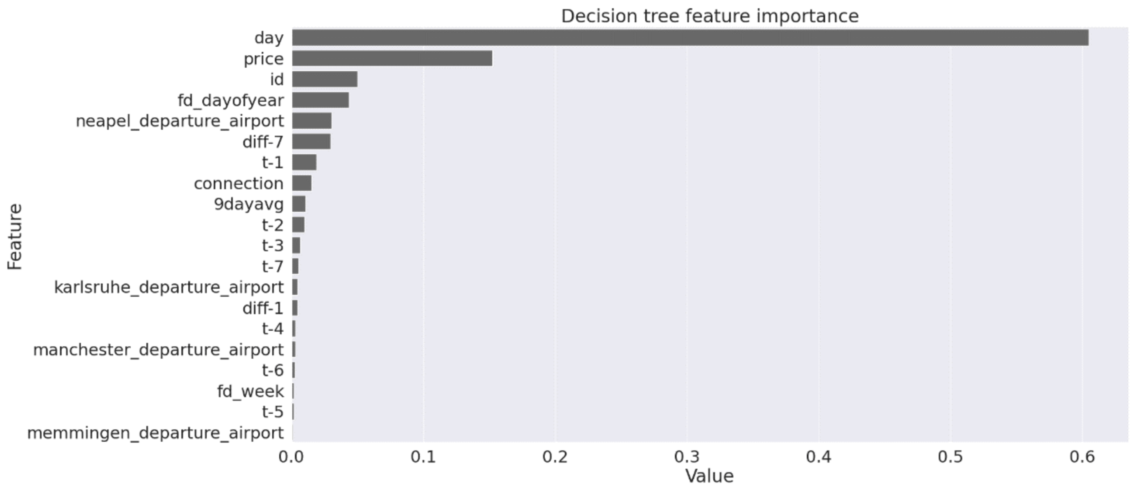


Figure 21: Feature Importance of the Decision Tree Model

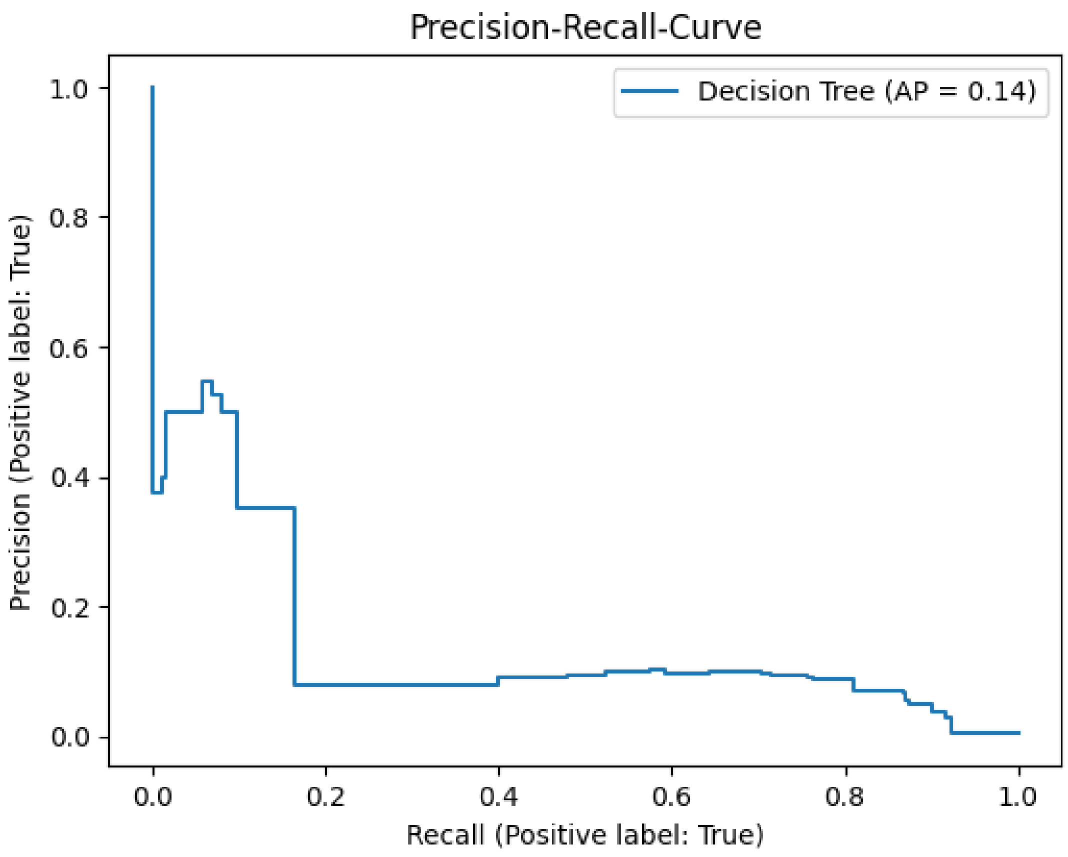
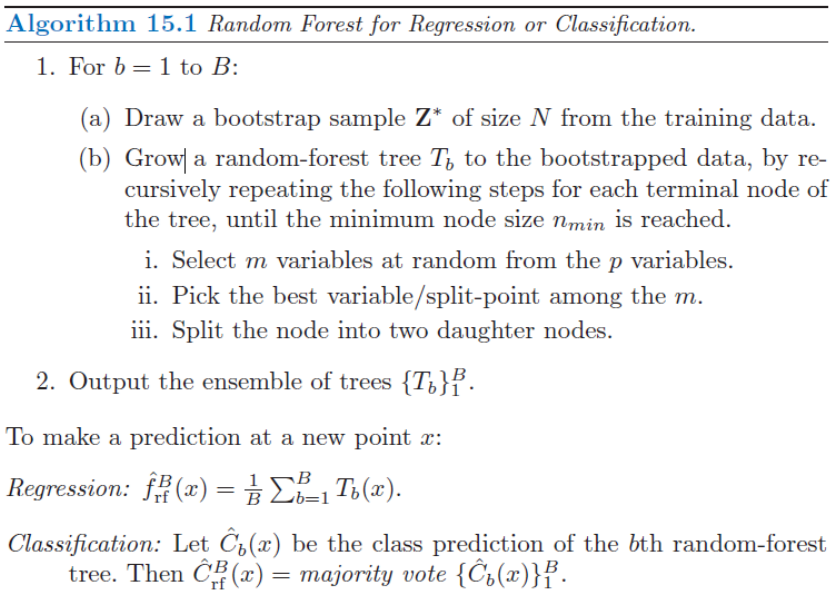


Figure 22: Precision-Recall-Curve of the Decision Tree Model

7.2 Random Forest

Random forests use two of the concepts mentioned above: bagging and the random subspace method. Random forest is an ensemble of similar decision trees. Decision trees, when grown deep enough, have low bias and a high variance according to (Hastie, et al., 2009, pp. 887-588). By creating a large number of trees with bagging and the random subspace method, and then averaging the predictions helps to decrease the variance and results in a model with low bias and low variance.



Algorithm 15.1 *Random Forest for Regression or Classification.*

```
1. For $b = 1$ to $B$:
    (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size $N$ from the training data.
    (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
        i. Select $m$ variables at random from the $p$ variables.
        ii. Pick the best variable/split-point among the $m$.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.
```

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Figure 23: Random Forest Algorithm (Hastie, et al., 2009, p.588)

In Figure 23 the random forest algorithm is presented. (Hastie, et al., 2009,p.593) describes the calculation of the feature importance like this: “At each split in each tree, the improvement in the split-criterion is the importance measure attributed to the splitting variable, and is accumulated over all the trees in the forest separately for each variable”.

7.2.1 Implementation

The hyperparameters for the random forest model are similar to the decision tree model. This produces also many very similar trees, which results only in a slight performance improvement over the decision tree model.

Hyperparameters (default value)	**Value**
n_estimators (100)	45
Split criterion (gini)	gini
max_depth (None)	8
min_samples_split (2)	2
min_samples_leaf (1)	22
min_weight_fraction_leaf (0,0)	0,0
max_features (None)	**None**
random_state (None)	None
max_leaf_nodes (None)	150
min_impurity_decrease (0,0)	0,0
bootstrap (True)	
oop_score (False)	
n_jobs (None)	None
random_state (None)	42
verbose (0)	0
warm_start (False)	False
class_weight (None)	None
ccp_alpha (0,0)	0,0
max_samples (None)	None

Table 6: Hyperparameter Random Forest

In the model only bagging was used but not the random subspace method. By limiting the number of features, the trees become significantly worse, because the features "day" and "price" have the highest predictive value. If the amount of features f is reduced to $\sqrt{f}$, the original amount of 43 features is limited to 7 in the subsamples. This means that the probability that one tree has one of the two best features is only at 30%. The other 70% of the trees have a much worse performance. This effect pulls down the predictive performance of the ensemble. To use the random subspace method there must be more usable features added or the number of useless features has to be reduced. But at least the precision-recall curve is smoother.

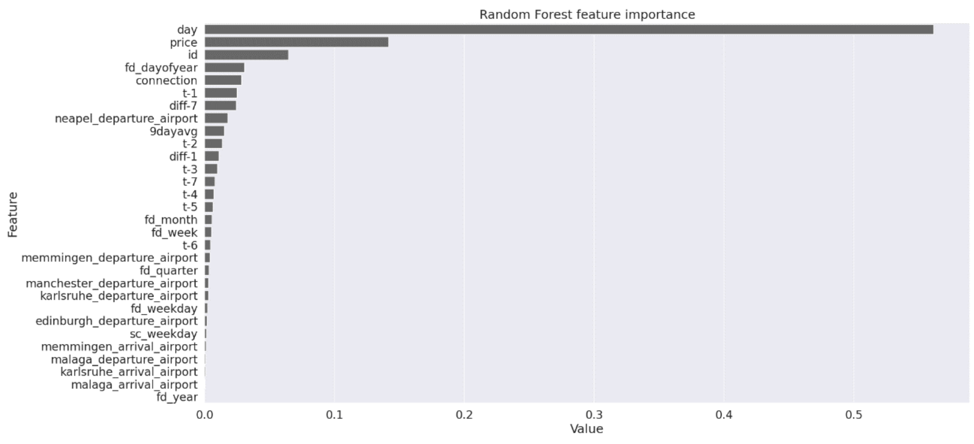


Figure 24: Feature Importance of the Random Forest Model

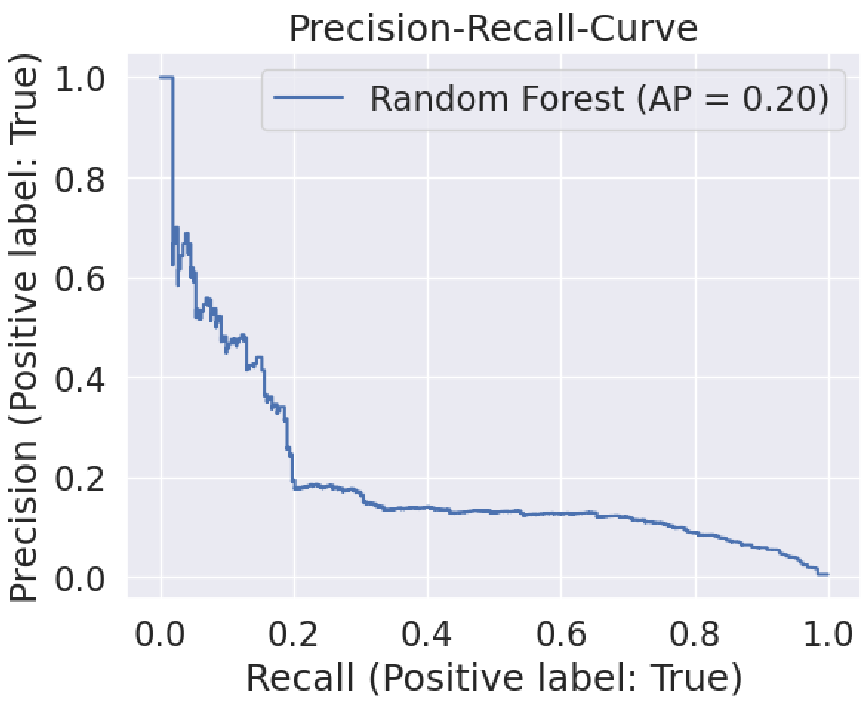
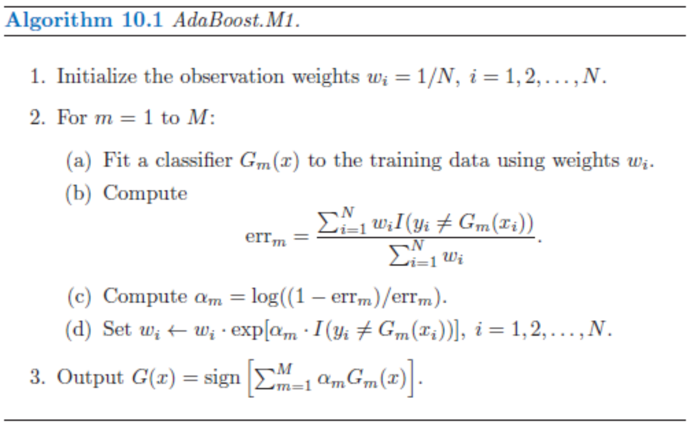


Figure 25: Precision Recall Curve of the Random Forest Model

7.3 AdaBoost

AdaBoost uses boosting techniques to increase the performance of weak classifiers. Most of the time, decision trees are used as classifier. After (Geron, 2022, p. 223), AdaBoost trains first a base classifier on the training set und uses it to make predicions on the training set. The weight of the misclassified instances is now increased. On this new training set with weighted instances a new classifier is trained and the process is repeated. This creates an ensamble of trees. For the final prediction all trees make predictions, which are weighted according to their accurancy on the training set.



Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
        $\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

Figure 26: The Algorithm AdaBoost.M1 (Hastie, et al., 2009, p.338)

In detail it works according to (Hastie, et al., 2009, pp.337-339) like this: There is a two class problem with an output variable $Y \in \{-1,1\}$. The input is defined as X and the classifier G(X) prediction is either -1 or 1. This results in an error rate on the training sample of: $\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$

The error rate of a weak classifier is only a bit above random estimates. If now M weak classifiers $G_m(x)\ with\ m = 1,2, \ldots, M$ are trained like random forests, they can all give an ensemble prediction. But this time the classifiers are weighted by a weight α.

Now back to the training. There is a second set of weights $w_i$for the training observations (x_i, y_i) with i=1, 2,..., N and N the number of observations. For the base model the weights are $w_i = {}^{1}/_{N}$, and the result a standard decision tree, if the model uses decision tree as classifier. The next step 2(a) shown in Figure 26 2(a) is the training of the base model. In 2(b) the error rate is computed. The weight α_m of model G_m is calculated in 2(c). In 2(d) the weights of the observation for the training of the model $G_{m+1}$are created.

After all models of the ensemble are created, the final prediction is

$$G(x) = sign(\textstyle\sum_{m=1}^{M} \alpha_m G_m(x))$$

The AdaBoost classifier has additionally to the base model hyperparameters only three more: the number of trees, the learning rate and the used algorithm. Usually AdaBoost is using decision stumps (one-level decision trees) as classifiers. But AdaBoost also works well with decision trees of a higher complexity.

7.3.1 Implementation

AdaBoost offer less model parameters. Usually the max_depth is set to one and the model is using only decision stomps, what makes the configuration easy. In the tests max_depth=2 gave a better validation score.

Hyperparameters (default value)	**Value**
estimator	DecisionTreeClassifier
n_estimators (50)	60
learning_rate (1)	1
algorithm (SAMME.R)	SAMME.R
Split criterion (gini)	gini
max_depth (None)	2
random_state (None)	42

Table 7: Hyperparameter AdaBoost

In Figure 27 the features “id” and “day” are contributing most to the model. But Ada-Boost is also using other features in a far greater amount. As we will see later, the AdaBoost performance is bad on the test set, but it has the highest generalization ca-pability.

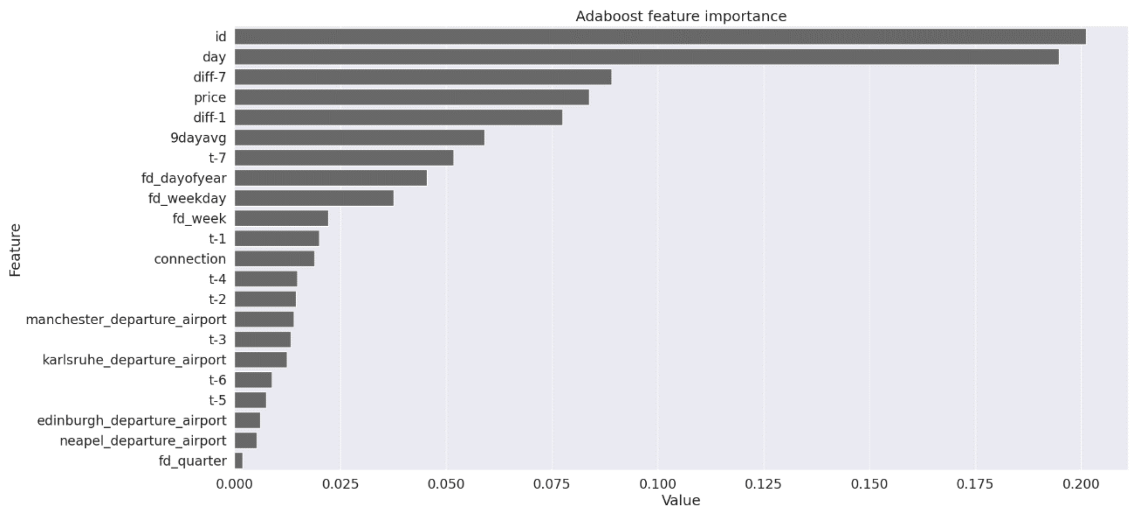


Figure 27: Feature Importance of the AdaBoost Model

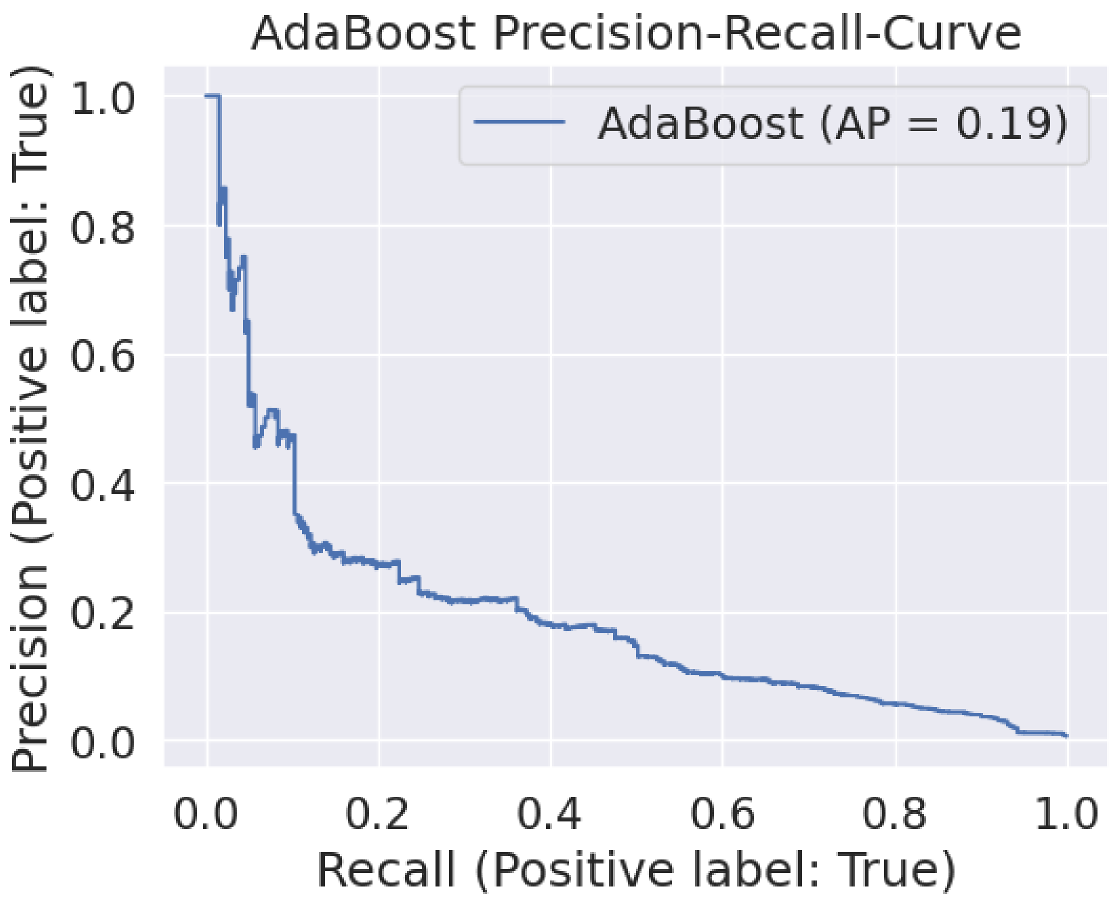


Figure 28: Precision Recall Curve of the AdaBoost Model

The precision-recall curve of AdaBoost looks also smoother, the average precision is with 0.19 similar to the random forest model.

7.4 XGBoost

Because of content constrains, gradient boosting will only be briefly explained. XGBoost was introduced by (Chen, August 2016) as a gradient boost algorithm, which adds sequentially predictors to an ensemble like AdaBoost. But this time the boosting is not applied with weights on the dataset. The boosting is now correcting the residual errors of the pervious predictors by predicting them.

The result is also an ensemble of decision trees. The learning rate scales the contribution of each tree.

7.4.1 Implementation

Hyperparameters (default value)	Value
scale_pos_weight	2
n_estimators (50)	70
learning_rate (1)	1
eta	0,4
max_depth (None)	10
random_state (None)	42

Table 8: Hyperparameter XGBoost

Like AdaBoost, XGBoost is using many different features, but still "day" is the dominating one.

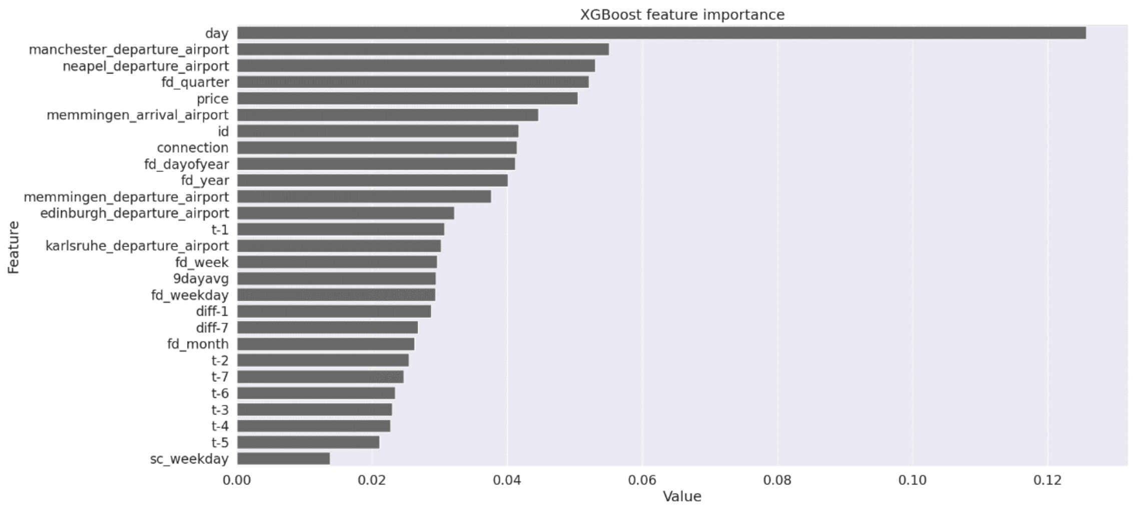


Figure 29: Feature Importance of the XGBoost Model

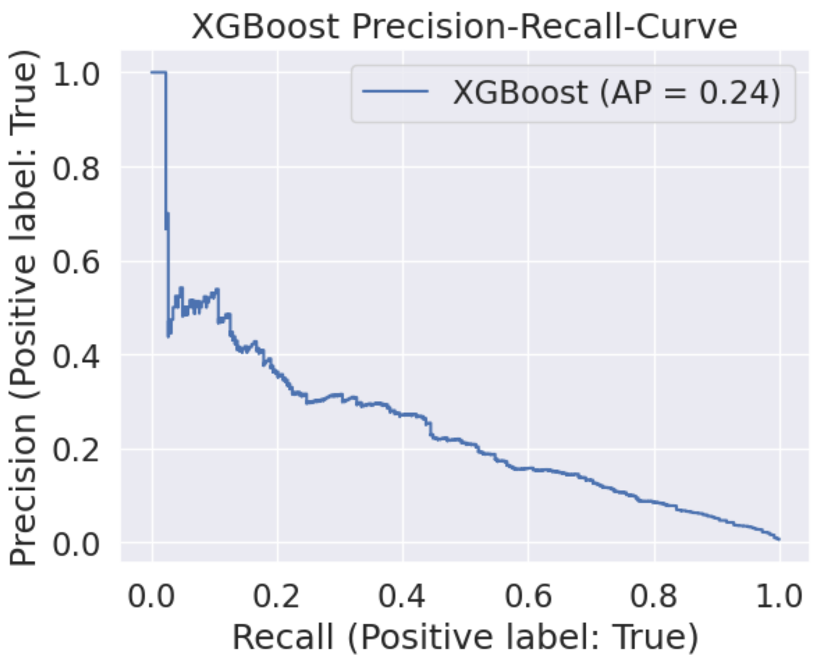


Figure 30: Precision Recall Curve of the XGBoost Model

7.5 Stacking and Voting Ensembles

Voting uses hard or soft vote and combines the prediction of several predictors to one final prediction. In Figure 31 we can see the prediction process of a voting ensemble. A new instance is first given as an input to the different ensemble models. Every model makes a prediction, which aggregates these to a final prediction by voting.

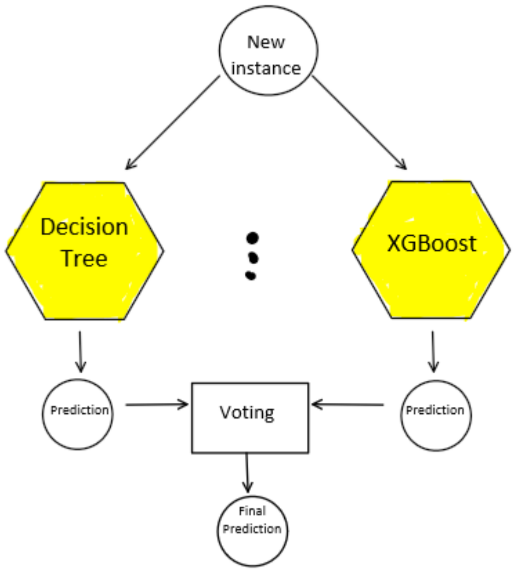


Figure 31: Voting Ensemble

Stacking Ensembles widen this approach by training a model instead using voting to perform the final prediction. Final predictor is called blender or meta learner, which can be seen in Figure 32. In a classification setting the blender can be any classification model like logistic regression, random forest or a XGBoost classifier.

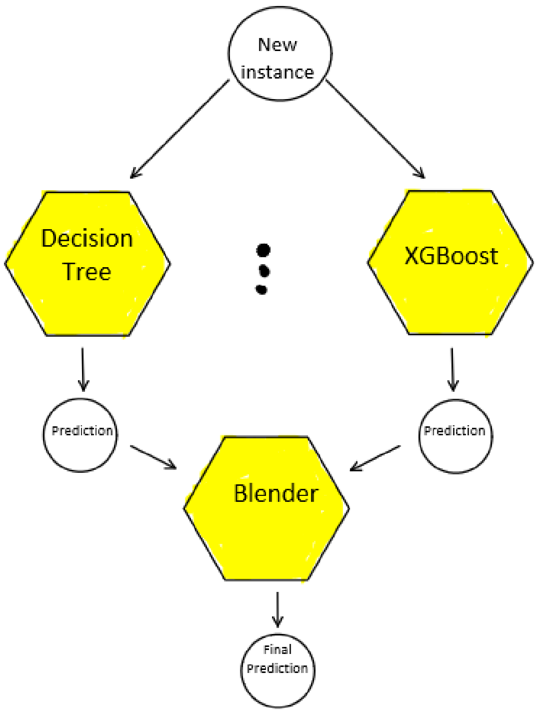


Figure 32: Stacking Ensemble

7.5.1 Implementation

Usually, the blender is trained on a specific hold out set, which is a subset of the training set, but was not used in training the individual models. For the ensemble models, the already pretrained decision tree, random forest, AdaBoost and XGBoost models with their optimized hyperparameters are used.

There are two voting classifiers using both hard and soft voting.

The stacking classifiers use three different blender models: a logistic regression, a decision tree and a random forest model.

7.6 Dummy Classifier

The predictions of the dummy classifier are not using the features as an input. We are using two different ones.

Dummy 1 always predicts the most frequent class, which is in our case "0".

Dummy 2 predicts randomly according to the class prior probabilities.

Both classifiers achieve a accuracy score of 0,99, because the dataset is highly imbalanced.

8 Evaluation

In this chapter we evaluate the predictions of the different models on test set and test set 2. We try to compare the models with each other on different aspects and metrics to choose the best model for our use case.

8.1 Evaluation on the Test Set

In Table 9 the trained models are now evaluated on the test set. Both dummy models have all scores close to zero, what must be expected with random guesses.

The $F_{0.5}$-Scores of the trained models are all close together with AdaBoost as the lowest performing model with 0,252. The basic decision tree classifier with 0,283 has a better performance. Most of the more complex models are slightly above. The model with the far best performance is the Soft-Voting classifier with 0,325.

Classifier	**Precision**	**Recall**	**F0.5 Score test set**	**F0.5 Score training set**	**MCC**
Dummy 1	0,000	0,000	0,000	0,000	0,000
Dummy 2	0,000	0,007	0,008	0,010	0,003
Decision Tree	**0,531**	0,099	0,283	0,395	0,221
Random Forest	0,470	0,118	0,294	0,396	0,233
AdaBoost	0,251	0,253	0,252	0,401	0,249
XGBoost	0,277	0,380	0,293	**0,991**	**0,320**
Stacking with Logistic Regression	0,249	**0,433**	0,273	**0,911**	**0,324**
Stacking with Decision Tree	0,288	0,334	0,297	**0,993**	**0,307**
Stacking with Random Forest	0,292	0,373	0,305	**0,993**	**0,321**
Soft Voting	0,349	0,255	**0,325**	0,664	0,297
Hard Voting	0,466	0,103	0,273	0,432	0,217

Table 9: Comparison of the different Models on the Test Set

Looking now on the precision and recall we can see high differences. In precision the decision tree classifier outperforms with 0,531 all other models but underperforms in recall. There on the other hand is the logistic regression stacking model the highest performing model with a recall of 0,433 (and the lowest precision score). The soft voting classifier has an average score in both precision (0,349) and recall (0,255)

The precision-recall curve in Figure 33 gives a different picture, but it is without the voting classifiers. The stacking classifier, the random forest and the decision tree model have a jumpy curve, while AdaBoost and XGBoost have a good performance over all thresholds. AdaBoost outperforms XGBoost only in some low recall/ high precision combinations, but overall, in this diagram XGBoost is performing the best.

If we look now on the MCC in Table 9 the XGBoost models and the Stacking Classifiers have the best score, while the decision tree, the random forest and the Adaboost model are outclassed by far.

The training time of the decision tree is low seen in Table 10: Training time in seconds. Random forest, AdaBoost and XGBoost have moderate training times. The stacking classifiers have also low training times, but they are using the already pretrained models. Without pretraining, the value in the brackets show the real training time, which is around the same as the voting classifiers.

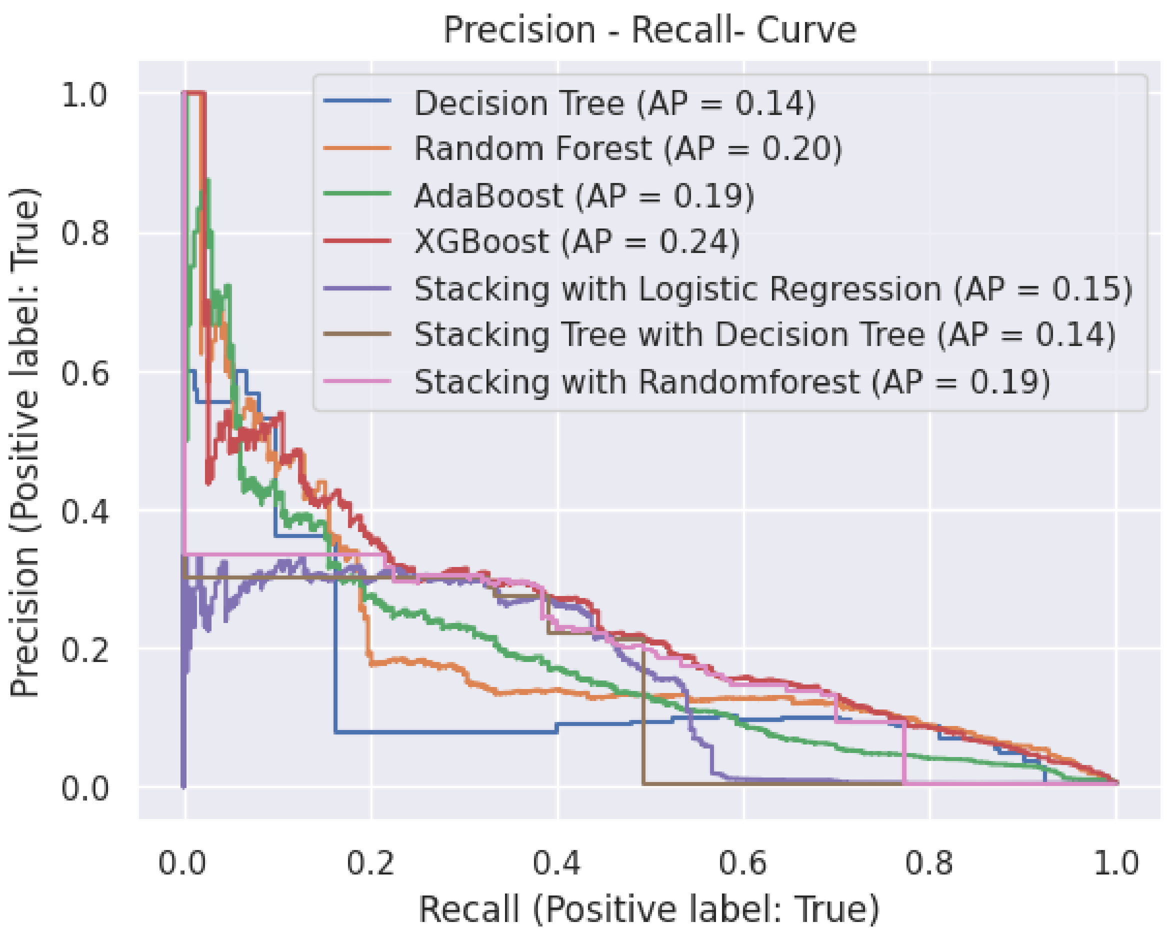


Figure 33: Precision Recall Curve of some Models on the Test Set

Classifier	**Training time in s**
Decision Tree	0,0129
Random Forest	471
AdaBoost	286
XGBoost	360
Stacking with Logistic Regression	52,2 (1169)
Stacking with Decision Tree	47,1 (1164)
Stacking with Random Forest	102 (1219)
Soft Voting	1122
Hard Voting	1108

Table 10: Training time in seconds

Based on the metrics there is not one best performing model. Overall, the soft voting classifier and the XGBoost model are overperforming, but also the simple decision tree delivers good results and should be taken into consideration, because it also provides speed and explainability.

If we now take the minimum requirements defined in the Problem Statement in chapter 1.3, we must admit that the minimum requirements of a $F_{0.5}$-Score of 0.6 is far away from the best score of 0,325 achieved on the soft voting classifier. Therefore, the models cannot be used as a decision support for buying cheap flights from Ryanair in the future. To increase the performance, it is probably recommended to increase the data base. More data, especially data from inside the company, would perhaps lead to much better results.

8.2 Evaluation on a Test Set with previously unknown Flight Connections

To test the ability of generalization, six of the models are now tasked to classify flight prices from flight connections, which are not included in the training set.

Classifier	**Precision**	**Recall**	$F_{0.5}$**-Score on test set 2**	$F_{0.5}$**-Score on test set**
Dummy 1	0,000	0,000	0,000	0,000
Dummy 2	0,014	0,007	0,012	0,008
Decision Tree	0,355	0,113	0,249	0,283
Random Forest	0,425	0,111	0,271	**0,294**
AdaBoost	0,335	0,179	**0,285**	0,252
XGBoost	0,282	0,265	0,278	**0,293**

Table 11: Comparison of different Models on Test Set 2

Here the model performances surprise again. The $\boldsymbol{F_{0.5}}$-Score of the decision tree decrease by 3 points, the random forest and the XGBoost model also perform worse on test set 2 than on the original test set. But the performance of AdaBoost, the worst model on the test set, outperforms the other models clearly.

8.3 Bonus: Threshold Optimization for further Improvement

In Table 12 the performance of the optimized models on the test set are quite disappointing. The AdaBoost and the XGBoost model performances change only slightly, but in both cases the new threshold is also close to the old one. In the case of the decision tree and the random forest the optimized threshold are with 0,356 and 0,304 far away from the standard 0,5. Despite all expectations the "optimized" models have a incredible low $F_{0.5}$-Score.

Classifier	$F_{0.5}$**-Score on test set**	**optimized** $\boldsymbol{F_{0.5}}$**-Score on test set**	**Threshold**
Dummy 1	0,000	0,006	0,000
Dummy 2	0,008	0,006	0,000
Decision Tree	0,283	0,120	**0,355**
Random Forest	**0,294**	0,150	**0,304**
AdaBoost	0,252	0,240	0,499
XGBoost	0,293	**0,302**	0,525

Table 12: Comparison of different Models with Optimized Threshold on the Test Set

In this case the threshold optimization doesn't provide a positive improvement and will probably not become a best practice in ML in this form.

9 Conclusion and Lessons Learned

Unfortunately, the best classifier doesn't fulfill the minimum requirements, but it outperforms random guesses by far. Classification with decision trees and ensemble methods in time series works in general, but for the use case the performance was not good enough.

Probably the models explain the planed price increases, but they don't predict the price increases caused by corrections in predicted demand by Ryanair. Data for this is not available outside the company and thus cannot be included in the model.

The algorithms by themselves showed inconclusive performances. The worst algorithm on the test set had the best score on the training set. Different metrics produced different best results. The selection of the right training set and the right metric influences greatly the final evaluation. And the threshold optimization only worked on the training set.

Lessons learned:

- Bad data quality causes issues during the whole process
- Changing features greatly influences the predictive quality of the models.
- If the initial objective is not viable, change the objective.
- Don't rely on free web-based tools, which also offer a premium version. Google Colab turns "busy" in the moments it is the most needed.
- While collecting data, store everything possible, it can be deleted later.
- Variable declarations are important. String operations are slow. Use small basic data types, integers instead of floats, small integer instead of big integers and booleans if possible. Save repeating strings in a separate table connected with an identifier.
- It is not recommended writing parts of the program in different languages and different styles. This causes confusion later on in the process.

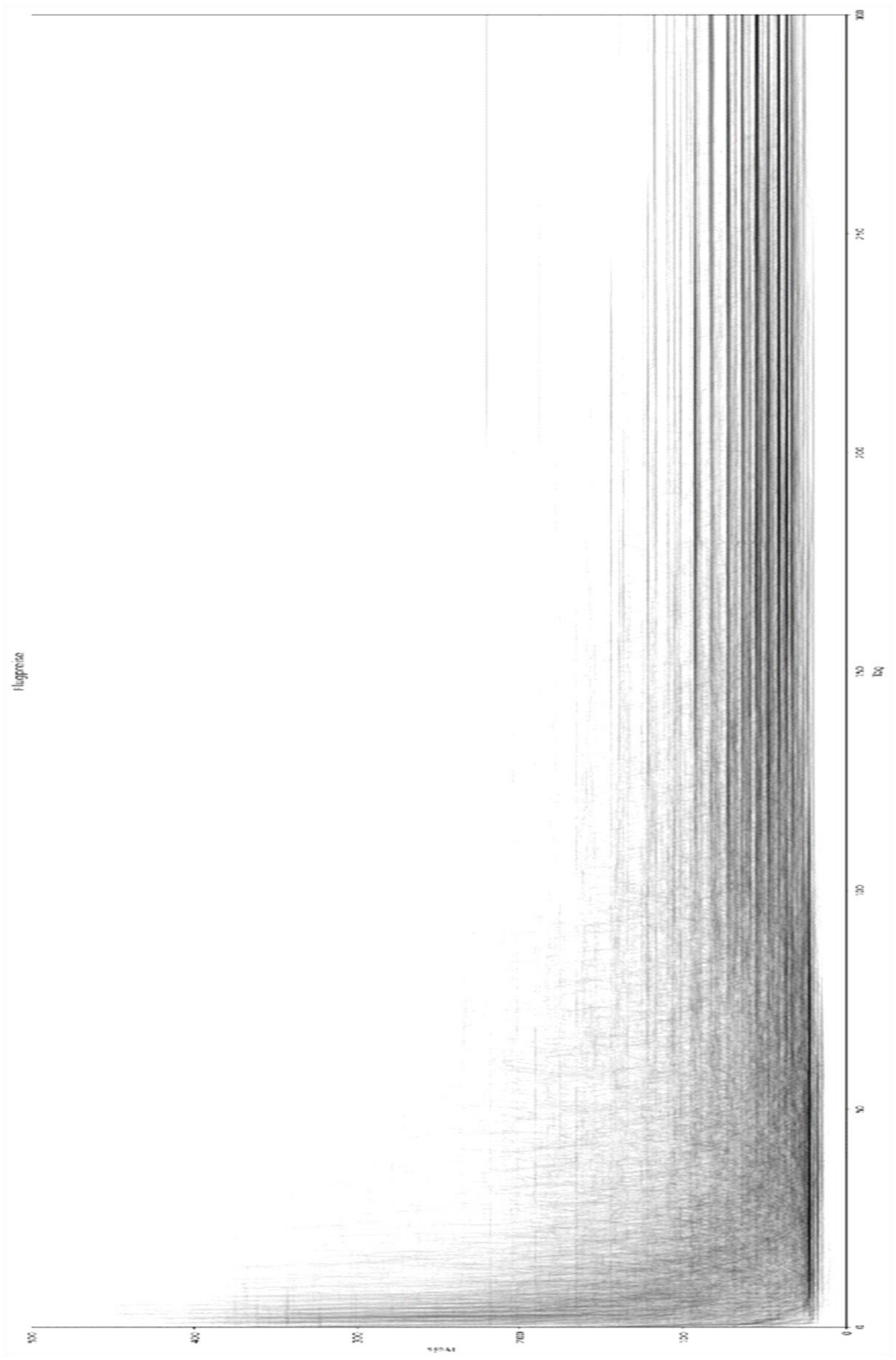


Figure 34: all flight prices dependent on the days of departure

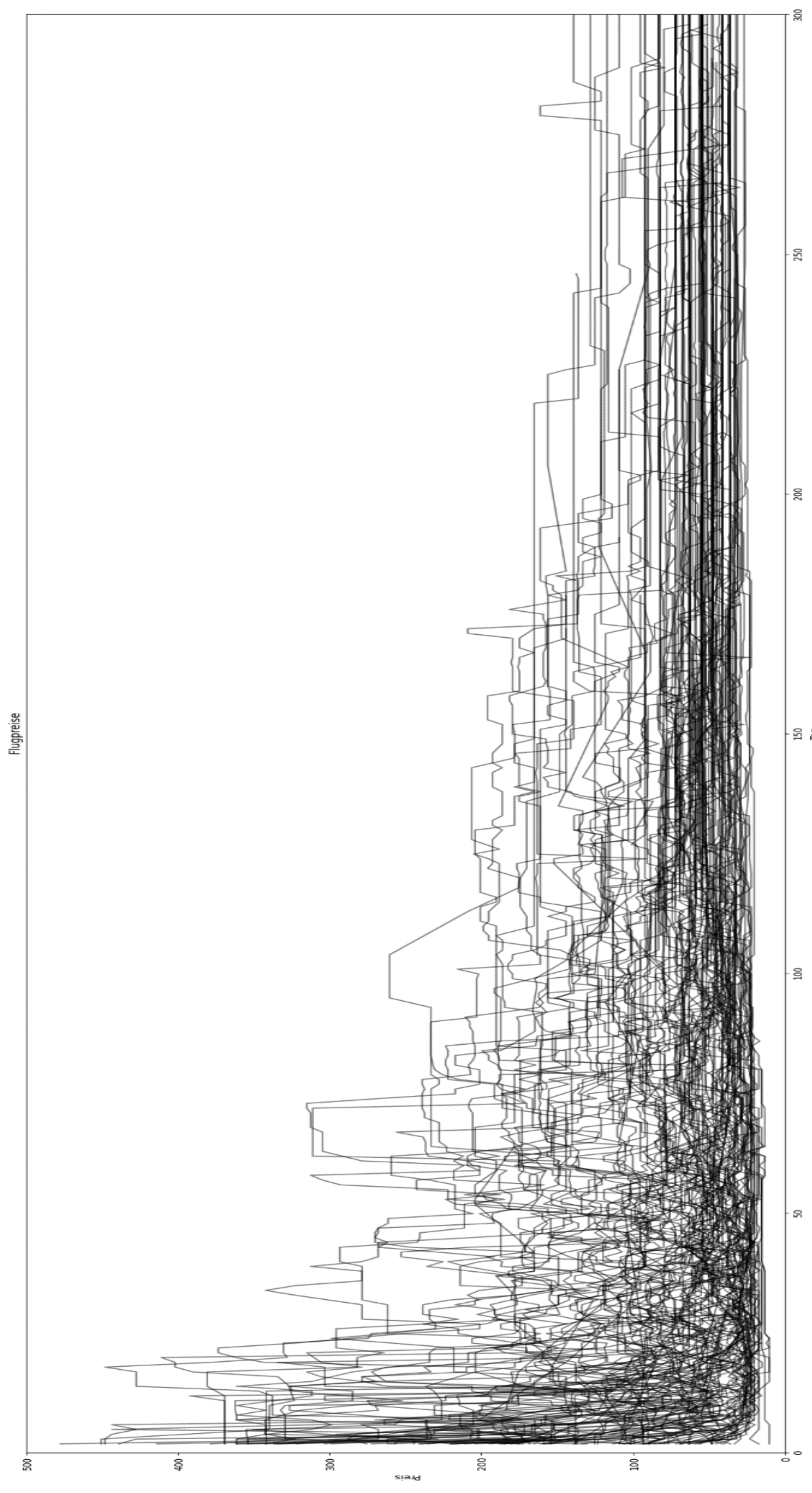


Figure 35: Price Progression of some Flights of 2022

References

Auffarth, B., 2021. *Machine Learning for Time-Series with Python.* Birmingham: Packt Publishing Ltd..

Breiman, L., 2001. Random Forests. *Machine Learning,* pp. 5-32.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J., 1984. *Classification and Regression Trees.* Boca Raton: Chapman & Hall/CRC.

Chen, T., August 2016. *XGBOOST: A Scalable Tree Boosting System.* s.l., the 22nd ACM SIGKDD International Conference.

Diller, H. & Herrman, A., 2003. *Handbuch Preispolitik.* Wiesbaden: Gabler Verlag.

Freund, Y. & Schapire, R. E., 1996. Experiments with a New Boosting Algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*.

Freund, Y. & Schapire, R. E., 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *journal of computer and system sciences,* pp. 119-139.

Friedman, J. H., 2021. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics,* pp. 1189-1232.

Geron, A., 2022. *Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow.* Sebastopol: O'Reilly Media.

Hastie, T., Tibshirani, R. & Friedman, J., 2009. *The Elements of Statistical Learning.* New York: Springer.

Ho, T. K., 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(8), pp. 832 - 844.

James, G., Witten, D., Hastie, T. & Tibshirani, R., 2013. *An Introduction to Statistical Learning.* New York: Springer.

Joseph, M., 2022. *Modern Time Series Forecasting with Python.* Birmingham: Packt Publishing Ltd..

Ke, G. et al., 2017. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* Long Beach, CA, s.n.

Kimms, A. & Klein, R., 2005. Revenue Management im Branchenvergleich. *Zeitschrift für Betriebswirtschaft, Ergänzungsherf 1 "Revenue Management",* pp. 1-30.

Klein, R. & Steinhardt, C., 2008. *Revenue Management Grundlagen und Mathematische Methoden.* Heidelberg: Springer- Verlag.

Makridakis, S., Evangelos, S. & Assimakopoulos, V., 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting,* pp. 1346-1364.

Makridakis, S., Spiloitis, E. & Assimakopoulos, V., 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *Internation Journal of Forecasting,* pp. 54-74.

Matthews, B. W., 1975. Comparison of the predicted and observed secondary structure of T4 pahge Iysozyme. *Biochimica et Biophysica Acta,* pp. 442-451.

McCarthy, S., 2022. *Ryanair Group Annual Report 2022,* s.l.: s.n.

Meffert, H., Burmann, C. & Kirchgeorg, M., 2008. *Marketing Grundlagen marktorientierter Unternehmensführung.* Wiesbaden: Betriebswirtschaftlicher Verlag Dr. Th. Gabler.

Nielsen, A., 2019. *Practical Time Series Analysis.* Sebastopol: O'Reilly Media.

Pedregosa, 2023. *Scikit-learn.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
[Accessed 30 7 2023].

Pedregosa, F. a. V. G. a. G. A. a. M. V. T. B. a. G. O. a. B. M. a. P. P. W. R. a. D. V. a. V. J. a. P. A. a. C. a. B. M. a. P. a. D., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research,* Volume 12, pp. 2825--2830.

Pfeifer, P. E., 1989. The airline discount fare allocation problem. *Decision Sciences,* pp. 149-157.

Shy, O., 2008. *How to Price.* Cambridge: Cambridge University Press.

Vinod, B., 2021. *The Evolution of Yield Management in the Airline Industry.* Cham: Springer Nature Switzerland AG.

Witten, D., Hastie, T., Tibshirani, R. & James, G., 2013. *An Introduction to Statistical Learning.* New York: Springer.

BLOCKING NOTICE

Ich erkläre, dass ich die Seminararbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich, inhaltlich oder sinngemäß entnommenen Stellen als solche den wissenschaftlichen Anforderungen entsprechend kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für Zeichnungen, Skizzen oder graphische Darstellungen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft und ausschließlich für Prüfungszwecke gespeichert wird.

Estepona, 02.08.2023 [illegible]

Ort, Datum Unterschrift