Forecasting flight prices with machine learning models: a comparative analysis between low and high-cost airlines

Sophia Daly

Dissertation written under the supervision of professor Pedro Afonso Fernandes

Dissertation submitted in partial fulfilment of requirements for the MSc in Business Analytics, at the Universidade Católica Portuguesa, September 2023.

Forecasting flight prices with machine learning models: a comparative analysis between low and high-cost airlines

Sophia Daly

Abstract

Forecasting flight prices is a challenging task due to the complex nature of the pricing algorithms that airlines use. Apart from the fact that these algorithms are not public, they have to take into account many different variables that affect ticket prices. Since the airlines' demand forecasting may not always hold true as a result of varying demand, prices need to be adjusted accordingly. This approach is called dynamic pricing. It is a technique of price discrimination based on temporal differences mainly, leading to the widely spread assumption that the time of booking is a crucial determinant of the ticket price. This analysis shows that apart from days to departure, especially flight distance and airline type influence the price significantly. That is, longer flights as well as flights operated by full-service carriers, as opposed to low-cost carriers, are usually more expensive.

This thesis uses a dataset including the flight fares and other flight-related characteristics of one-way flights in the US between April and October 2022, retrieved from the search engine *Expedia.com*. The data is used to train and compare the performance of several supervised learning models aiming to forecast flight prices. Each model is deployed three times, first with the entire dataset, and then once with data only from low-cost-carrier and only from full-service-carriers, respectively. The most accurate models for all three datasets are the random forests followed by k-nearest-neighbor. The results of this thesis suggest that a large part of the flight price can be predicted using flight-related details such as days to departure and flight duration, yet, it also shows that there remains a certain inexplicable variability that could be due to external factors that are not included in the present analysis.

Keywords: Price prediction; dynamic pricing; machine learning; airline industry; random forest.

Forecasting flight prices with machine learning models: a comparative analysis between low and high-cost airlines

Sophia Daly

Resumo

Prever os preços de voo é uma tarefa desafiante devido à natureza complexa dos algoritmos de fixação de preços que as companhias aéreas utilizam habitualmente. Para além da sua natureza privada, estes algoritmos levam em consideração muitas variáveis diferentes que afetam, por essa via, os preços das passagens aéreas. Uma vez que a previsão da procura pelas rotas das companhias aéreas nem sempre se mantém válida devido à sua variabilidade ao longo do tempo, os preços precisam de ser ajustados continuamente de modo a favorecer a rentabilidade dessas companhias. Esta prática designa-se por fixação de preços dinâmica, uma técnica de discriminação de preços baseada principalmente em diferenças temporais, levando à amplamente difundida perceção de que o momento da reserva é o principal determinante do preço das passagem aéreas. A presente análise revela que, para além do número de dias até à data de partida, o tipo de companhia aérea e, sobretudo, a distância de voo também influenciam significativamente o respetivo preço. Assim, voos mais longos e operados por companhias de serviço completo, em oposição às companhias de baixo custo, são geralmente mais caros.

A presente tese utilizou uma base de dados incluindo os preços das passagens aéreas e outras características relacionadas com voos de ida nos EUA entre abril e outubro de 2022, obtidas através do motor de busca *Expedia.com.* Estes dados foram utilizados para treinar e comparar o desempenho de vários modelos de aprendizagem automática supervisionada com o objetivo de prever os preços de voo. Cada modelo foi implementado três vezes, primeiro com a base de dados completa, depois com os registos relativos às companhias de baixo custo e, finalmente, apenas com os dados das companhias de serviço completo. Os modelos mais precisos para os três conjuntos de dados são as florestas aleatória seguidos pelos modelos de K vizinhanças próximas. Os resultados deste trabalho sugerem que uma parte significativa do preço pode ser prevista utilizando detalhes relacionados com o voo, como o número de dias até a partida e a duração da viagem. Contudo, permanece uma certa variabilidade não explicada que pode dever-se a fatores externos não incluídos na presente análise.

Palavras chave: Previsão de preços; fixação dinâmica de preços; aprendizagem automática; setor da aviação; florestas aleatórias.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and objectives

Nearly everyone that travels regularly has likely already faced the following decision: should I book my flights early out of fear that prices will rise closer to the travel date and I will not lose my seat or should I rather wait in the hope that prices decrease as part of some last-minute offer? Both scenarios are realistic and it is quite difficult to make an informed decision as a consumer (Groves and Gini, 2015; Rajankar and Sakharkar, 2019).

The common perception when purchasing flight tickets is that the earlier the tickets are bought, the cheaper they are (Lantseva et al., 2015; Groves and Gini, 2013). This suggests that days to departure is the key predictor for flight price. Recent research, however, has shown that this is not always true in reality (Lantseva et al., 2015; Groves and Gini, 2013). The days to departure are doubtlessly an important factor in flight pricing, nevertheless, other variables might play an important role as well. Furthermore, low- and high-cost airlines follow different pricing strategies (Kwoka et al., 2016; Etzioni et al., 2003), so one could assume that the price determinants differ from one airline type to the other.

The biggest challenge in predicting flight prices is the information asymmetry between customers and airlines (Gordiievych and Shubin, 2015; Groves and Gini, 2015; Rajankar and Sakharkar, 2019; Wang, 2020). A customer only has very limited knowledge, namely publicly available information such as flight date and flight route, about a certain flight of interest. Yet, airlines determine their prices based on highly complex pricing mechanisms involving many variables that are not known to the customer, like number of available seats, for instance. Obviously, these pricing algorithms are not public, making it hardly possible for consumers to correctly anticipate the price changes. However, analyzing pricing patterns and determining the most influential predictors can help customers make more appropriate choices about when to purchase and what price to expect (Groves and Gini, 2013; Wang et al., 2019).

The objective of this thesis is to investigate whether flight prices can be predicted accurately solely with information regarding the flight characteristics. This will be done by comparing the performance of several different machine learning models with selected performance measures. To get more detailed insights, each model will be done three times, first for the total dataset, then only for low-cost carriers, and lastly only for full-service carriers with the goal to find the best model for each airline type.

1.2 Research question

To what extent are flight details sufficient for predicting flight prices? What are the most influential determinants of flight prices and do they differ between airline types?

Moreover, what is the best machine learning model to predict airfare prices?

1.3 Scope of analysis

The present thesis intends to identify the most important factors influencing flight prices and compare whether these are the same for low- as well as high-cost airline types. Moreover, different machine learning models will be compared with the ultimate goal of predicting flight fare prices. In terms of scope, the focus lies specifically on the comparison between low- and high-cost airlines operating in the United States.

The remainder of the thesis is divided into the following parts: Section 2 presents an introduction to the airline industry and the different airline types. Additionally, current and past literature in the area of flight price forecasting is summarized and different machine learning approaches are discussed. Section 3 provides a summary of the models used in the course of this work including the chosen measures to compare the models' performances. After that, the dataset is presented in section 4, while section 5 explains the modelling process and section 6 and 7 conclude the analysis with the presentation and discussion of the findings, as well as the limitations and need for future research.

2 Background and Related Work

2.1 Airline types

Airlines can be divided into low-cost carriers (hereafter LCC) and full-service carriers (hereafter FSC), differentiating themselves by operational characteristics and generic strategy. While LCC prioritize cost leadership, offering a form of transportation with inexpensive fares, FSC offer a higher-quality product with superior service level (Rozenberg et al., 2014).

LCC manage to offer lower prices, which is their main marketing goal, due to several reasons. Firstly, they have product homogeneity by only flying regular flights, not having charter flights and usually only flying a single type of aircraft resulting in lower maintenance costs and higher productivity. Additionally, LCC generally do not have connection flights, flying only point-to-point, which reduces the time on the ground and allows to increase the number of flights per day. Apart from that, LCC focus merely on leisure flights, they do not offer first or business class tickets and usually fly to secondary airports where the airport charges are lower. Lastly, the standard fees do not include any extra services such as food and drink, insurance or baggage. If these services are desired, then an additional price has to be paid.

The FSC, on the other hand, focus on high quality and differentiation. Services such as re-booking possibilities, snacks and drinks during the flight and more comfortable seating are usually included in the ticket fare. As opposed to LCC, FSC offer an extensive route network due to their cooperation with other carriers and the *hub and spoke* system, where several primary and secondary airports are connected allowing for further trips (Coto-Millán et al., 2015).

Given the two distinct business models of LCC and FSC the main competitors of the airlines are within the same category of airlines. Nevertheless, some scholars suggest that LCC increasingly become direct competitors across categories as well (Kwoka et al., 2016). This is due to the fact that as they operate on main routes more and more, and move from secondary airports to the main airports they become more dominant and with that, a direct threat to all the other established airlines. With respect to the resulting effect on the ticket prices, the authors claim that LCC strongly influence other LCC' prices as well as FSC, while the latter only marginally affect other airlines in their category and have no impact on LCC. This outcome suggests taking the number of competitors into account when analyzing the determining factors of flight prices.

2.2 Pricing mechanism

The reason for the complexity in forecasting flight prices is the fact that many stakeholders, such as airlines, competitors, airports, customers, play a role in the final price.

Varying demand, time of booking, competitors' moves, time of the year and external factors such as fuel prices are reasons why prices for tickets are dynamic. Since the airlines' strategies and forecasts of demand may vary as a result of the interplay of these factors, it would not make sense for an airline to sell their tickets for a static price, and therefore dynamic pricing or yield/revenue management (these terms will be used interchangeably in this thesis) is used as their pricing mechanism.

Dynamic pricing is "the study of determining optimal selling prices of products or services, in a setting where prices can easily and frequently be adjusted" (Boer, 2015). Ultimately, it is a profit maximization strategy that can be used when certain characteristics are given. Firstly, there exists only a fixed amount of inventory and any unsold inventory is lost, it cannot be sold at a later point in time and is not replenishable. Secondly, the product must typically be sold long in advance and demand can fluctuate significantly. Lastly, a further important condition is low marginal sales costs and high production costs (Kimes, 1989). In the case of airline tickets the inventory is the available seats in an airplane, that cannot be sold once the flight takes off (an empty seat is forever lost revenue). Tickets are usually available for sale several months before the departure date and demand can vary largely due to various reasons. Also, the airline industry has high fixed costs and low, highly variable marginal sales costs. Therefore, revenue management can be applied to flight ticket sales. In fact, Donaghy et al. (1995) state that revenue management was developed by the airline industry as a consequence of the deregulation in 1978 and is a mechanism mainly used in the hotel and airline industry.

Dynamic pricing is essentially a form of price discrimination since different prices are charged from different people at different points in time. Price discrimination can happen due to temporal, spatial or income differences between customers. In the case of flight tickets, the temporal difference is the main driving factor, assuming that the customers' willingness to pay (WTP) increases closer to the flight date. This results in a *low-before-high-arrival* pattern, meaning people with lower WTP buy their tickets longer in advance than those who are prepared to spend more (Selcuk and Avşar, 2019). This may also be one of the reasons why airlines charge higher prices closer to the departure day. Carter (1988, as cited in Donaghy et al., 1995) further explains that (instead of reducing fares prior to flight departures, yield management looks at historical demand patterns and identifies seats which have been difficult to sell in the past. These seats are promoted often through discounting to ensure they are sold in advance, resulting in the last available seats on a flight being available only at full fare). That being said, it can also happen that prices are decreased shortly before the flight to stimulate demand, in case the desired number of tickets have not been sold yet.

2.3 Flight price forecasting

Research around flight pricing and optimal purchase timing has been conducted for various objectives in the past. The topic has been addressed both as regression as well as classification problem, depending on whether the intended output was a decision such as *buy* or *wait*, for instance, or a continuous outcome like the actual flight price. Given the complexity of airlines' pricing methods, it is a challenging task and therefore several approaches have been already tested.

Most importantly, a good forecast relies on the extraction of relevant features in the data (Groves and Gini, 2013). Although different models use different sets of features, one of the most commonly mentioned factor influencing the ticket price is the remaining days to departure (Groves and Gini, 2015; Etzioni et al., 2003; Rajure et al., 2021; Malighetti et al., 2015), supporting the hypothesis that purchasing as early as possible is the best way to go. Additionally, the number of remaining seats, ticket class, competitors' prices and day of the week have been found to influence ticket prices significantly (Rajure et al., 2021; Groves and Gini, 2011; Malighetti et al., 2015).

A pilot study investigating two round-trip routes in the US with 12,000 observations over a time period of 41 days has been conducted by Etzioni et al. (2003). The goal was to distinguish pricing patterns within the data, compare different models for analyzing these patterns and eventually find out whether it is possible to save money as a consumer applying the proposed models. The predictors used in the models were flight number, hours to departure, current price, airline and route. In terms of methods, the authors applied the Ripper Rule Learner, Q-learning, time series and combined each of their models through stacking. The ensemble model output was a *buy* or *wait* decision and led to correctly indicating possible savings 61.8% of the time.

Apart from that, it was found that there are commonly price changes 7 and 14 days before departure but also that prices tend to fluctuate more during certain times of the year, e.g. holidays. Moreover, airlines were separated into two categories. The first including FSCs while the latter was made up LCC. The division into the groups was based on the finding that airlines of a similar type usually follow similar pricing patters. That is, FSC airlines sell their tickets for higher prices and fluctuate often, whereas the LCC show less fluctuation and lower fares. The main drawback of this pilot study by Etzioni et al. is the fact that the data used did not have any indication on how many seats were still left when a certain price was observed.

Similarly, Rajankar and Sakharkar (2019) published a paper aiming to predict flight prices with a dataset considering the features origin, destination, departure date and time, arrival time, total fare, airline and flight date for their models. The authors proposed various algorithms for predicting flight prices including supervised models (Decision Tree, Random Forest, K-Nearest Neighbour, Support Vector Machine and Gradient Boosting

Regression) and a neural network, the Multilayer Perceptron (MLP). The evaluation metrics used for the paper were R-Squared, Mean Absolute Error (MAE) and Mean Squared Error (MSE) whereby the random forest outperformed all other models with a R^2 value of 0.67 and a MAE and MSE of 0.08 and 0.04, respectively. The MLP performed almost equally well resulting in metrics very similar to the random forest. All the other models fall behind significantly having lower R^2 results and higher errors.

Furthermore, the low-cost airline EasyJet, one of the main LCC in Europe, has been thoroughly studied with regard to their pricing strategy, development and effects on competitors' prices (Malighetti et al., 2015). The paper states that when LCC were firstly introduced, purchase timing was the main differentiating factor between prices and that up to 90% of the final price could be saved when booking at the earliest possible date. However, concerning price determinants nowadays the authors found out that purchase timing is not the only important causal factor of price. Day of purchase and departure time also have an effect on ticket price. The highest prices were registered between Saturday and Monday, while the lowest fares were on Wednesday and Thursday and flights departing in the afternoon were typically the most expensive. Moreover, flight distance also affects the price in the way that the average fare per kilometre is lower and more stable for longer distances compared to shorter flights. An additional point of interest in this paper is the role of direct and indirect competition on fare prices. The presence of competition on a certain route directly influences EasyJet's market share, and consequently affected ticket price as well. The more competitors operating on the same route, the lower the prices. The results of this study were found by applying a linear regression model including time-variant and time-invariant predictors.

Groves and Gini (2011) proposed a partial least square regression model for predicting the best time to purchase airline tickets. Their goal was to predict expected minimum future prices including an estimate of the risks of price changes. They analyzed data starting 60 days prior to the departure date, generated an automated optimal feature set selection and added time-delayed observations to the feature vector. Certain recurring patterns, and therefore structured price volatility, could be found in the data. Especially the day of purchase was an influential factor for the quoted price. From Tuesday to Thursday prices were lower than at the weekends, which is in line with the finding of the EasyJet case study described above.

Additionally, price changes varied depending on the route. Apart from the fact that having more competing airlines influenced prices, the nature of the flight, meaning leisure or business flights, affect the pricing patterns. The assumption presented is that flights during the week, in many cases business flights, are less price-sensitive, since one specific flight at a specific time must be taken regardless of the price.

The model was also compared to a similar mechanism by BingTravel, which equally intends to give recommendations about whether to wait or buy flight tickets right away.

The model by BingTravel results in a *buy* decision much more frequently, at least 70% of the time, than the model by Groves and Gini, which does so under 20% of the days. Also, it the BingTravel model only considers the following 7 days, therefore not taking into account longer time frames.

The study concludes by stating that it is possible to predict ticket prices with publicly-available information and achieve significant savings when purchasing at the right time, again emphasizing the importance of the right purchase timing, which is not necessarily the earliest possible date. Also, in a subsequent study the same authors used a another multivariate regression and classification procedure, and, once again, it appears that it is not always the cheapest option to purchase tickets as early as possible (Groves and Gini, 2015).

Papadakis (2012) takes a slightly different approach with regard to the feature selection of his predictive models. He assumes that the flight price is made up of two categories of variables, those affecting the base price, and those affecting price fluctuation. In his paper, he focuses only on the variables influencing fluctuation, using only days to departure and recent price as features, and compares the performance of three models, them being Ripple Down Rule Learner, Logistic Regression and Linear Support Vector Machine (SVM) with the goal to predict flight price. The Logistic Regression and the Linear SVM had similar accuracy results (69.9% and 69.4%, respectively), while the Ripple Down Rule Learner outperformed both with an accuracy of 74.5%. Yet, a major drawback of this study is the relatively small dataset consisting of only 720 data points.

Flight price forecasting has not only been tackled with machine learning methods, but also with classical statistical approaches such ARIMA (Auto-regressive Integrated Moving Average) model (Gordiievych and Shubin, 2015). The authors state that it is not necessary to consider all the price-determining factors, such as number of seats remaining, but rather only to focus on predicting the likelihood of price in- or decrease by looking at a few factors in particular. The paper does not report on the exact results of the model but state again that buying earliest possible is not always the best strategy.

3 Methodology

Forecasting flight prices, as the name suggests, aims to predict the price of a specific flight, an unknown value (the response or output variable), by analyzing values of known variables in the data such as flight departure and arrival airport and days remaining to departure. This task will be done with the use of Machine Learning algorithms. Machine learning is a part of Artificial Intelligence (AI) and describes "the collection of methods for extracting (predictive) models from data" (Provost and Fawcett, 2013).

Machine learning can further be divided into several subcategories depending on the nature of the data and the problem at hand. The three most important categories are supervised learning, unsupervised learning and reinforcement learning. Supervised learning covers all the problems in which the input, the predictors, as well as the output, the response, are known (labelled data). Based on these inputs and outputs, an algorithm will learn to relate the output variable to the input variables by continuously minimizing the difference between the predictions made by the model and the actual value. As opposed to supervised learning, unsupervised learning would be used in cases in which the output is not known, i.e. unlabelled data, and the algorithm will try to find patterns in the data, without knowing what exactly to predict (James et al., 2021; Friedman, 2006).

The present dataset contains labelled data, so all the algorithms used for this thesis are supervised learning algorithms. Furthermore, the output variable, the flight price, is continuous, so it is a regression problem.

That being said, it is important to note that supervised learning is not the only possible approach to tackle this topic. To give an example, as was mentioned previously, Etzioni et al. (2003) used Q-Learning, a form of Reinforcement Learning, to predict whether the price will in- or decrease and consequently supply a *buy* or *wait* output.

The models examined in this thesis to forecast flight prices are linear regression, KNN (K-nearest neighbour), Gradient Boosting model, decision tree and random forest.

3.1 Proposed Models

3.1.1 OLS

The first proposed model is linear regression, using ordinary least squares (OLS) to fit the model. Linear regression is a simple, yet widely used method assuming a linear relationship between the predicting variable(s) x and the response variable y (James et al., 2021). Mathematically, a linear regressions can be written with the following formula:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_n x_{in} + \epsilon_i \tag{1}$$

where β_0 is the model intercept and β_1 to β_n determine the average effect of a unit increase in the independent variables x_1 to x_n on the dependent variable y, under the condition that all other predictors remain unchanged. In order to estimate the coefficients as accurately as possible and to find the regression line that best represents the relationship between the predictors and the outcome variable, the ordinary least square approach is applied, aiming to minimize the *residual sum of squares* (RSS). RSS represents the sum of all squared residual errors, that are calculated by the difference between the observed value and value predicted by the model:

$$RSS = \epsilon_1^2 + \epsilon_2^2 \cdots + \epsilon_n^2 = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 \tag{2}$$

Since predicting flight prices requires the detection of complex patterns in historical data, linear models might not lead to the more accurate results. Thus, it is necessary to apply different forms of supervised learning models (Wang et al., 2019).

OLS assumes a certain relationship between the predictors x and the outcome variable y and fit the models according to these assumptions, known as parametric analysis. In contrast, the following algorithms are non-parametric, which implies that the relationship between x and y will be discovered by the algorithm progressively and they are therefore more flexible (James et al., 2021).

3.1.2 K-Nearest Neighbour

The K-Nearest Neighbour (KNN) Regressor is a distance-based algorithm, making predictions for a specific data point depending on the weighted average of the K data points closest to the desired prediction point x_0 in the training data. In other words, instead of making predictions based on a mathematical function, the algorithm remembers the training data and applies the patterns to the test set.

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in N_0} y_i \tag{3}$$

The selection of the value for K plays an important role in the model's performance on the test data. When K takes on small value, the model becomes more versatile in fitting the data, leading to a low bias but high variance. Conversely, a high K value decreases variance but increases bias resulting in a less adaptable yet smoother fit. Choosing the ideal value for K is therefore subject to the *variance-bias trade-off* (Taddy, 2019).

3.1.3 Decision Tree

A decision tree can predict qualitative and quantitative outcomes. As a result, it can be used for regression, as well as classification problems (James et al., 2021; Taddy, 2019). Decision trees use a technique known as *recursive binary splitting*, meaning that input data is successively divided into non-overlapping regions R_1 to R_J, i.e. nodes (or leaves) according to certain features/splitting rules, each represented by a branch. These rules can be represented in the following formula, where j is the predictor at cutpoint s:

$$R_1(j,s) = \{X \mid X_j < s\} \quad \text{and} \quad R_2(j,s) = \{X \mid X_j \geq s\} \tag{4}$$

For each region a single prediction value/class will be computed, which is the mean value or mode of the output in the observations in the training data. Similar to the regression described previously, the splitting is done with the goal to minimize the output's RSS in the case of a regression tree:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R1})^2 \quad + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R2})^2 \tag{5}$$

At the end of each branch, following a number of splits, a certain output value or class in the case of a regression tree or classification tree, respectively, is estimated, which will be the forecasting result. There is a variance-bias trade-off between small and large trees. A decision tree can become very complex if it is split many times, resulting in many branches and leaves. While this will lead to very accurate results in the training data, it will most probably not perform as well on testing data, due to overfitting. To reduce the risk of overfitting, trees can be *pruned*. Just like pruning a biological tree, pruning a tree in machine learning essentially means eliminating certain parts, making it less complex and more generally applicable. This can be done by introducing the *cost of complexity*, a tuning parameter, which is typically the number of terminal nodes or leaves. Which parts are kept or removed is derived from a K-fold cross validation, aiming to minimize the out-of-sample classification error (Varian, 2014). In this analysis the decision trees will not be pruned, since the random forest algorithm, as described below, will be applied to all the dataset as well, which equally reduces the risk of overfitting for tree based models.

A big advantage of the decision tree is its easy interpretability, partly because of the possibility to visually represent the model in a two-dimensional graphic independently of the number of the characteristics of the dataset. Additionally they are relatively robust against irrelevant variables since these will simply not be chosen for splitting and finally the decision tree can be implemented easily without having to decide on numerous tuning parameters (Friedman, 2006).

3.1.4 Gradient Boosting Regression

The Gradient Boosting (GB) Regressor belongs to the category of boosting algorithms, a subset of ensemble learning techniques. Ensemble models are algorithms that combine the outcomes of several weak models to create a more powerful one. Boosting models are built by training new models over and over again, where each new model focuses on addressing and eliminating the residuals of the previous model. This iterative process continues until a certain criterion is met. The final prediction is obtained by combining the predictions of all the models and weighing each of them by their individual contribution.

The weak learners used in a GB Regressor are usually decision trees. The algorithm starts with a single leaf which represents the first guess for the output variable. If the intention is to predict a continuous value, which will be the case in this thesis, this first value will be the average value. Based on the errors of the first guess, being the differences between the predicted and the actual target values, a tree is created. Then, another tree is built intending to optimize the previous tree by reducing the errors. This procedure continues until either a defined number of trees has been reached, of any additional trees do not improve anymore. Also, the size of the trees can be predefined. According to Friedman, it typically lies between 4 and 10. To prevent overfitting, the weight of each tree in the final prediction is scaled by the *learning rate* (or *shrinkage factor*), a value between 0 and 1 (Friedman, 2006).

3.1.5 Random Forest

Decision trees typically perform very well on training data due to their ability to create complex structures with branches. As a result, they easily overfit leading to a good model fit on the training data but a much poorer performance whenever faced with new data. To mitigate this issue the random forest algorithm can be applied. Similar to the GB Regressor, the random forest is an ensemble learning method that combines many individual decision trees into a "forest". In the case of a regression problem, the algorithm takes the average of they individual trees' outcomes to create a final prediction. The random forest minimizes overfitting by selecting a random feature subset for each tree, therefore ensuring that the individual trees are not correlated with each other. Ultimately this leads to a more robust and more reliable output of the resulting average of the models (Taddy, 2019; James et al., 2021; Breiman, 2001).

3.2 Performance measures

The performance of the different models is assessed by how close the predictions are compared to the actual values. In order to compare the respective performances various metrics can be used.

3.2.1 MSE and RMSE

The first performance measure to be introduced is the *Mean Squared Error* (MSE). The MSE compares the predicted outcome values with the actual values by averaging the squared differences of all observations. Squaring the values results in them being positive. At the same time, large errors boost the value of the MSE, leading to a higher mean error. The lower the MSE, the better the model's performance.

$$MSE = \frac{RSS}{n} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

Typically, the test MSE is higher than the training MSE, since applying least squares pushes to minimize the RSS. This can lead to overfitting the model to the training data, which results in a worse performance on the test data. Also, when interpreting the results, it must not be forgotten that the unit now is also squared.

In order to get the error measure in the original unit, the square root of the MSE can be calculated. The error values stay positive, yet they match the unit of the target variable, making them more intuitive to comprehend and interpret.

$$RMSE = \sqrt{MSE} \tag{7}$$

3.2.2 MAE

As alternative to the MSE and the RMSE, the *Mean Absolute Error* (MAE) calculates the mean of the absolute differences between predicted and actual outcomes. As for the MSE, a lower MAE suggests a good model fit. An advantage of the MAE is that this measure is not as sensible to large error values as the MSE. Also, as for the RMSE, the target variable's unit remain unchanged when calculating the MAE.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

3.2.3 R-Squared

Lastly, the R-squared statistic is a metric specifying how much variance of the dependent variable is explained by the independent variables. The values lie between 0 and 1, with values closer to 1 indicating a better fit. R-squared is calculated by the subtracting the RSS from the total sum of squares (TSS), and subsequently dividing by the TSS (James et al., 2021).

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \tag{9}$$

4 Data

The dataset used in the analysis for this thesis was retrieved from *Kaggle* [1], which is eligible for public use according to the Attribution 4.0 international (CC BY 4.0) license. The data shows the prices listed for one-way flights on the travel search engine *Expedia.com* between April and October 2022 from and to 16 different airports in the United States. The original dataset contains 27 attributes including search and flight date, flight duration, price with and without tax, departure time, departure and arrival airport and number of seats remaining, to name a few. Each row represents one flight on a specific search date. The full list of features can be found in Appendix A.1.

4.1 Data pre-processing

The data pre-processing was done in RStudio, while the models were implemented in Python. Given the huge size of the original dataset (31.09 GB), it was firstly filtered to only one single search date, namely April 22, 2022. This date was chosen due to the following reasons: The observations in the dataset start on April 16, 2022, which was just a few days before Easter. It can be assumed that the observations on the first search days in the dataset do not represent the actual price patterns due to the fact that it is a long weekend for many people, and therefore airlines might decide to adjust their prices accordingly (Lantseva et al., 2015; Rozenberg et al., 2014). In order to exclude this potential distortion of the real pricing patterns, it was decided to use Wednesday after Easter, supposing that the prices on that day were not affected by any external factors. After that, flights with layover were removed for the simplicity of the analysis.

Subsequently, it was necessary to change several data types and create new variables. After casting the data types, the variable *dtd* (days to departure) was created by calculating the difference between the flight date and the search date. Moreover, *segmentsDurationInSeconds* was turned into *flightDurationMin* in order to display the travel duration in minutes, which is easier to read and interpret. Additionally, the departure time was extracted from the column *segmentsDepartureTime* into the column *dep_time*. Then, the departure times were divided into the four categories morning (6-12h), afternoon (12-18h), evening (18-24h) and night (0-6h).

Furthermore, the dataset was filtered to only include *coach* as cabin type, as the comparability was not given due to the low number of observations in the other categories. For the same reason, the dataset was filtered to exclude basic economy fares.

Based on the literature research conducted for this thesis, there is evidence that the flight prices are influenced by the weekday on which the flight is operated (Malighetti et al., 2015). To find out whether this is true in the present dataset, an ANOVA test

[1] `https://www.kaggle.com/datasets/dilwong/flightprices`

was conducted. ANOVA was the appropriate test as the goal was to find out whether there is a difference in mean total fare between the seven days of the week, that is, a comparison between several groups. Before conducting the test, a new column, namely *weekday_fl* was added, indicating on which day of the week the flight departs. The result of the ANOVA test shows that there is a significant difference in means between the weekdays at a 99% significance level. To get a more detailed insight, the Tukey's test was done, which compares all the days in a pairwise manner. The outcome of this comparison shows a significant difference in means for every pairwise comparison. The only pairs that show slightly less significant differences are Saturday-Monday, Wednesday-Tuesday and Thursday-Friday. Based on these results it does not make sense to group the days of the week into weekend and during the week.

Similarly, the relationship between the different airlines and total fare was tested using the same procedure. Again, a significant difference between the airlines and total fare was found. Additionally, the pairwise comparison implies a significant difference between all the airlines except *Delta Airlines* and *Alaska Airlines.*

Next, the dummy variable *lcc* (low cost carrier) was created indicating whether the operating airline is considered a low cost carrier or a full service carrier (hereafter LCC and FSC, respectively). The categorization was based on the grouping by Tsoukalas et al. (2008) resulting in three LCC (*Spirit Airlines*, *JetBlue Airways* and *Frontier Airlines*) and four FSC (*American Airlines*, *Delta Airlines*, *Alaska Airlines* and *United Airlines*). When comparing the LCC to FSC with regard to total fare, the t-test shows a significant difference between the two airline types, supporting the assumption that there is a notable price different between LCC and FSC.

Ultimately, all the unnecessary columns were removed resulting in a final dataset with 85 115 observations and 12 features. The pre-processing in R and the full feature list of the final dataset can be found in Appendix A.2 and A.3 respectively.

4.2 Exploratory Data Analysis

After the afore-mentioned pre-processing steps the dataset is ready to be examined in more depth. To start with, an exploratory data analysis will help to better understand the characteristics of the data at hand.

Initially, the main variable of interest of this analysis was the flight ticket price (*total-Fare*). The values range from 23.97 USD to 1665.60 USD. As indicated by the literature review and confirmed by the ANOVA test, the departure day significantly influences the total price. To get a first impression of the relationship between weekdays and flight prices, the average price per weekday is plotted in Figure 1. While Tuesday and Wednesday appear to be the cheapest flight days, flying on Sunday costs the most. Notably, this is true for low-cost, as well as full service carriers.

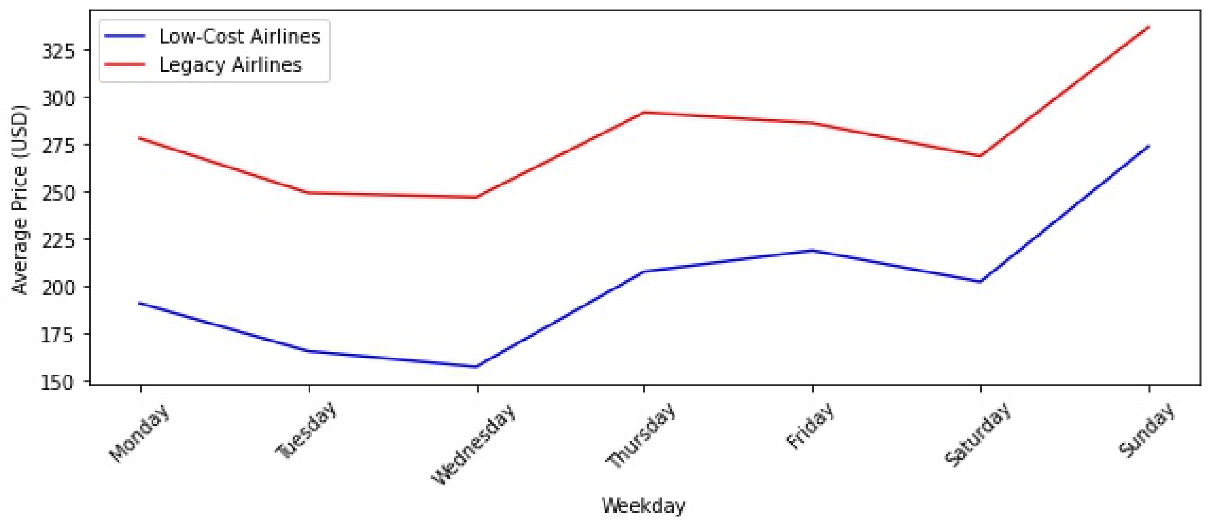


Figure 1: Average flight price per flight day

When having a look at the distribution of price values, it becomes apparent that they are positively skewed due to occasional tickets with extremely high prices compared to the rest. Applying a log transformation results in a more normal distribution (see Figure 2). This step is helpful since some models, such as Linear Regression, assume a linear relationship between the predictors and the outcome variable, which is is enhanced with the log transformation.

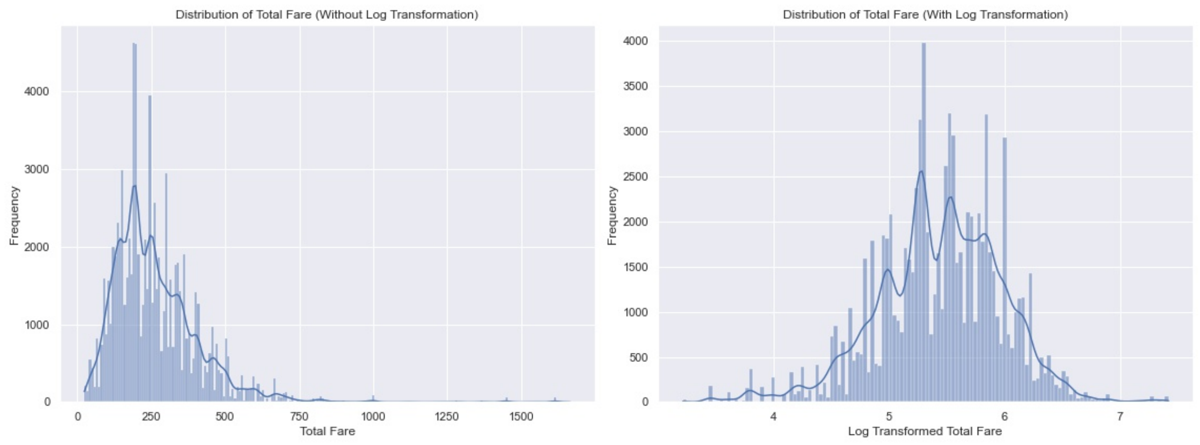


Figure 2: Distribution of total fare with and without log transformation

Apart from resulting in a more normal distribution of the output variable *total fare*, the log transformation leads to a partial elimination of the outliers and consequently reduce the influence of extreme values. As can be seen in Figure 3, some airlines have various tickets with fares significantly higher than their median values. Figure 4 shows the price distribution after the log transformation, due to which the distance between outliers and the median were notably reduced. Therefore, from now on the variable of interest will no longer be *totalFare*, but *totalFare_ log*.

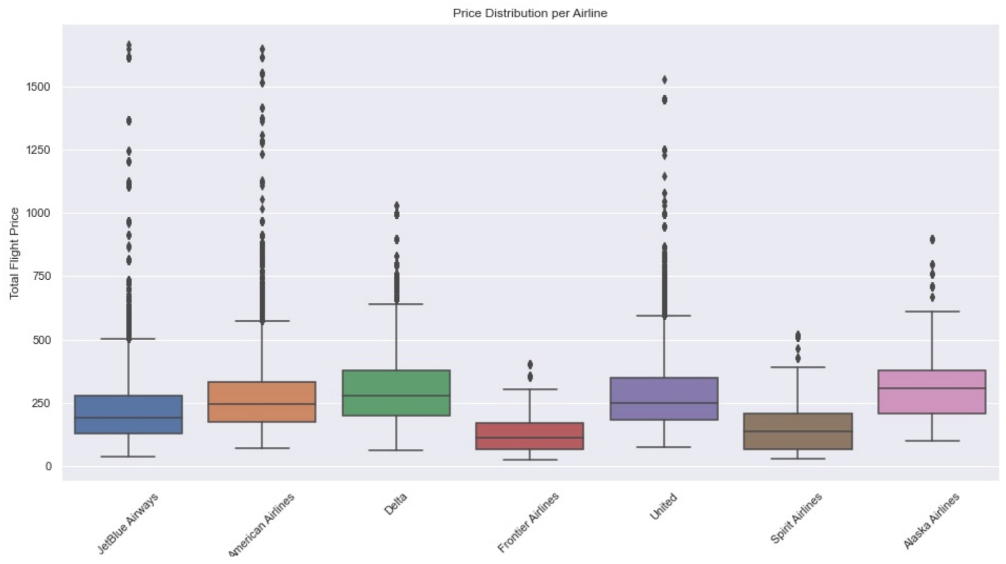


Figure 3: Price distribution per airline before log transformation

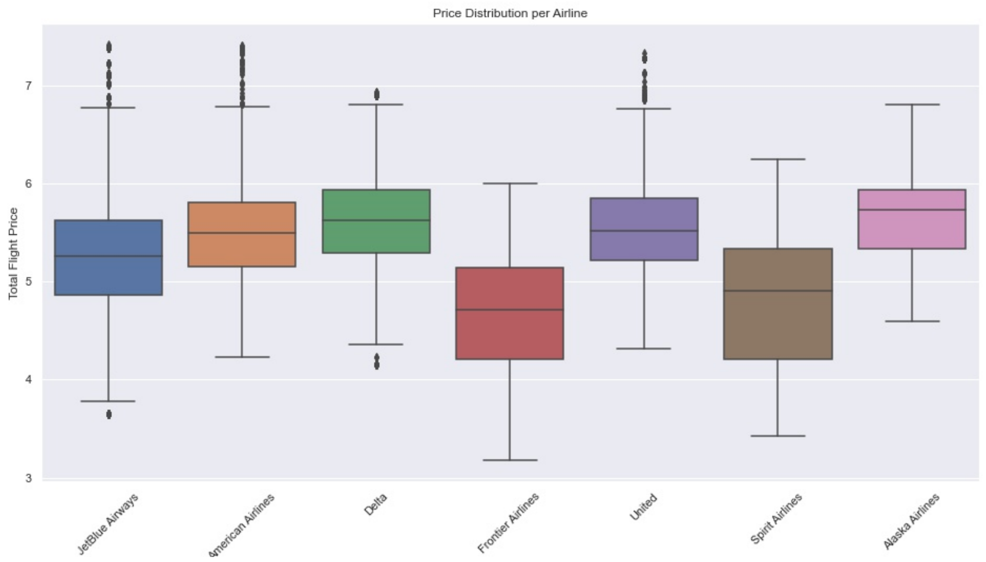


Figure 4: Price distribution per airline after log transformation

When comparing the two correlation matrices (Figure 5) one can see that, on the first glimpse, the feature correlations vary slightly in their magnitude, yet, the general directions are the same for both low cost and full service carriers. In both cases, total fare increases when days to departure decrease. This finding was to be expected and is in line with what other researches have found. Interestingly though, it seems that total fare also increases with a higher number seats remaining. These two features and their relationship to the total fare will be discussed in more detail below. Within the dataset *totalTravelDuration* is the only feature that has NAs. However, since travel distance and travel duration are closely correlated, the feature *flightDurationMIN* will be used as an indicator for the travel length.

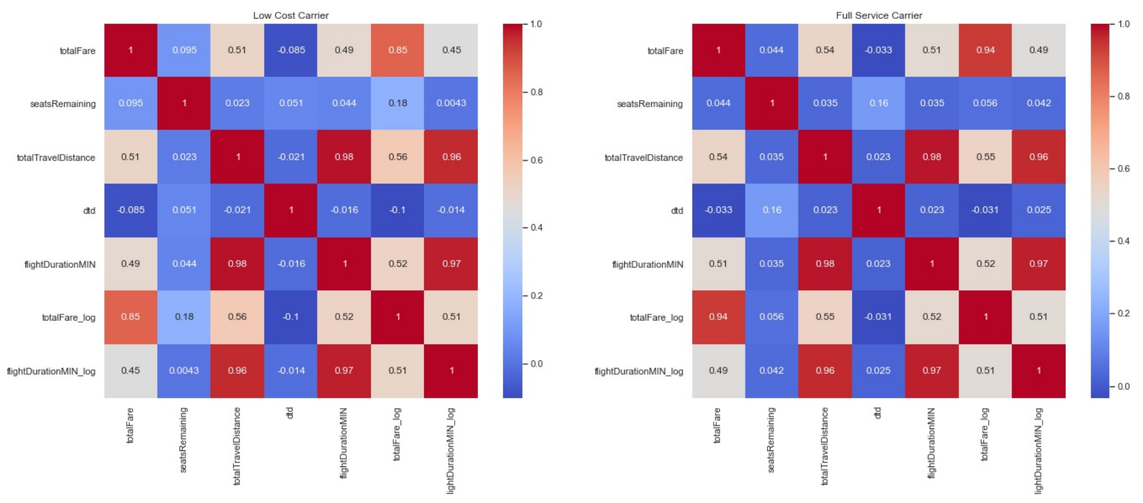


Figure 5: Correlation matrices low cost carrier vs legacy carrier

Figure 6 visualizes the relationship between days to departure and the average total fare by airline type. It shows that both low-cost and legacy carriers increase their prices one week before departure and charge the highest price four days before departure. After that, the prices seem to decrease until the day of the flight. This might be because the airlines want to fill up the planes and prefer selling the tickets for a lower price than not at all (Donaghy et al., 1995).

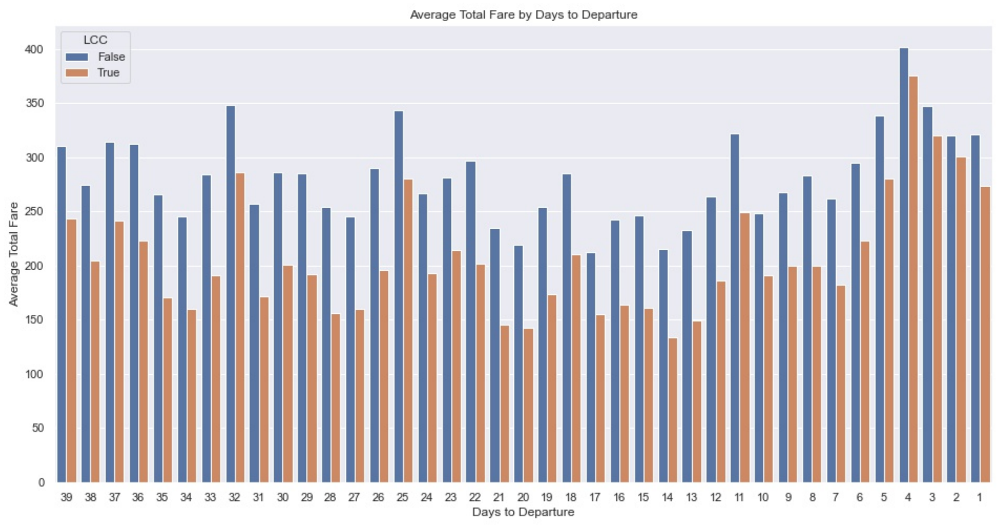


Figure 6: Average total fare by days to departure

Depending on the aircraft type, planes have different capacities with regard to how many passengers they can accommodate. One of most used aircrafts in the US in 2022 was the Boeing 737-800, which can hold 162-189 passengers [2]. The present dataset only includes observations with ten or less seats remaining. In contrast to almost 200 available seats, having only 10 remaining seats means the flight is almost fully booked. Looking at Figure 7, it seems like legacy carriers are not influenced a lot by the number of seats remaining. The average fare stays relatively constant while the number of seats remaining decreases. Low-cost carriers, on the other hand, fluctuate much more. While their average price for flights with 10 seats remaining is very low compared to the legacy carriers, it increases the fewer available seats there are. With one seat remaining the LCC even ask for a higher price than the legacy carriers. Lastly, and at first sight surprisingly, there seem to be observations with zero seats remaining that still have a price. A possible explanation might be overbooking.

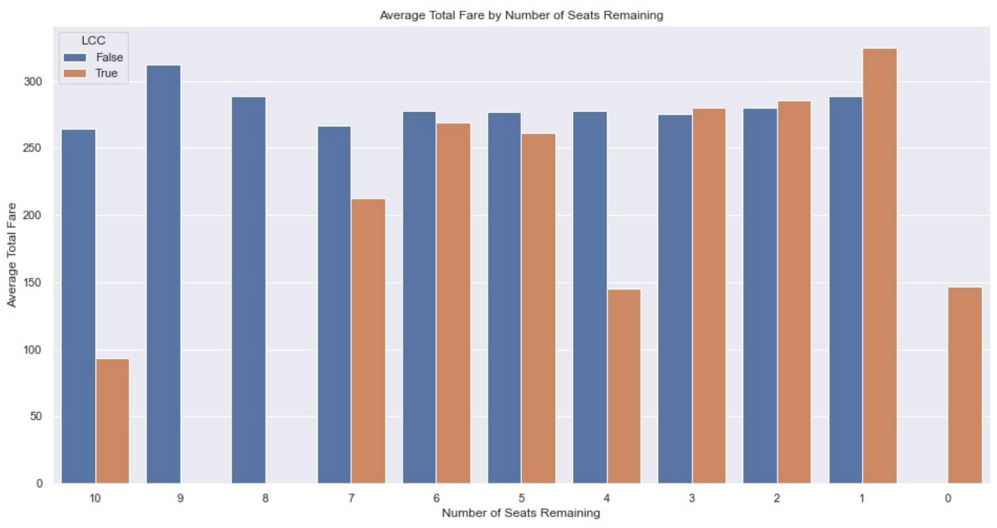


Figure 7: Average total fare by number of seats remaining

With respect to the allocation of observations between LCC and FSC, the two pie charts provided in Figure 8 illustrate that the majority (77%) of the observations represent full service carriers, whereas the remaining portion, 23%, belong to low-cost carriers. This distribution is relatively close to the real-word scenario where LCC made up around 30% of air traffic in 2016 (Kwoka et al., 2016).

5 Modelling

This section outlines the process of implementing several predictive models, including the necessary pre-processing steps involved and comparing their individual performances.

[2] `https://www.boeing.com/commercial/737ng/`

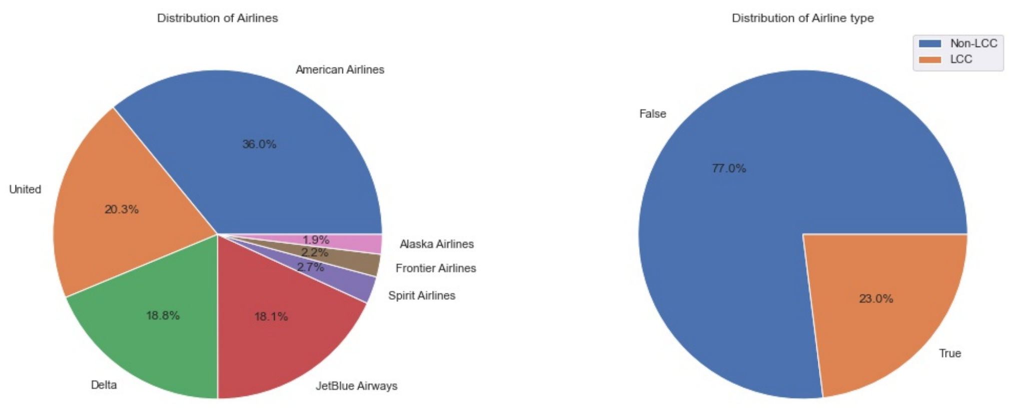


Figure 8: Distribution of airlines and airline type

The primary objective is to get a clearer understanding of the underlying patterns within the data and identify the most significant factors influencing flight price.

To start with, two further data sets were created in addition to the total dataset, one for low-cost carriers (*lcc = 1*) and another one for full-service carriers (*lcc = 0*) consisting of 19 606 and 65 506 observations, respectively. Then, all three data sets were split into training and test data using an 80/20 split ratio, separating the predictive features from the target variable *totalFare_log*.

In section 4.2 the advantages of the log transformation of the variable *totalFare* was already discussed. Apart from *totalFare*, the distribution of the feature *flightDuration-MIN* also showed an improvement from a log transformation and was closer to a normal distribution. For the remaining numerical variables no meaningful improvements could be observed, hence, the log transformation was not applied to any further features.

Subsequently, the categorical variables were one-hot encoded, converting them into dummy variables that represent all the different categories in each variable. In other words, non-numerical data was transformed into a numerical format that algorithms can deal with more effectively. The categorical variables that were one-hot encoded are *week-day_fl*, *segmentsAirlineName*, *day_time*, *startingAirport* and *destinationAirport*.

Following the one-hot encoding, some variables were *scaled*. Scaling (or *data normal-ization*) transforms the values of numerical features into a similar scale, in order for each variable to have the same weight in the model. Feature scaling is especially helpful for algorithms that do not assume a specific distribution of the data and are distance-based, such as KNN. There are several different scaling methods, which one is applicable depends on the characteristics of the data. For the present dataset, the MinMaxScaler was chosen. This method converts all the values into a range between 0 and 1 by applying the following formula:

$$X_{scaled} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \tag{10}$$

After the above-mentioned procedures, the different models were constructed. Each algorithm was trained and evaluated three times: once with the entire dataset, once with only LCC data, and once only with FSC data. By doing so, the models can effectively be compared based on their predictive performance not only with other models, but also their own performance with different datasets. The table below lists each model and the respective performance measures and computational time for each dataset.

Model	Dataset	MSE	MAE	RMSE	R-squared	Comp. Time (s)
LR	Total	0.1333	0.2811	0.3651	0.5695	0.57
LR	LCC	0.1885	0.3341	0.4341	0.5337	0.04
LR	FSC	0.1117	0.2603	0.3342	0.5076	0.13
KNN	Total	0.0831	0.2125	0.2882	0.7180	14.12
KNN	LCC	0.1203	0.2443	0.3468	0.7024	0.70
KNN	FSC	0.0742	0.2045	0.2723	0.6732	6.85
GB	Total	0.1062	0.2521	0.3258	0.6397	18.91
GB	LCC	0.1283	0.2705	0.3581	0.6827	2.80
GB	FSC	0.0877	0.2314	0.2961	0.6135	17.48
DT	Total	0.0913	0.1916	0.3038	0.6867	1.84
DT	LCC	0.1339	0.2183	0.3660	0.6686	0.26
DT	FSC	0.0844	0.1880	0.2904	0.6282	1.06
RF	Total	0.0500	0.1553	0.2236	0.8303	148.06
RF	LCC	0.0733	0.1796	0.2708	0.8186	25.12
RF	FSC	0.0441	0.1500	0.2100	0.8057	97.40

Table 1: Performance Metrics

The first model that was tested is the linear regression. As mentioned before in this thesis, the linear regression has a rather straight-forward way of predicting, which is also relatively simple to interpret. However, due to the simplicity of the model and the assumption that the predictors and the target variable share a linear relationship, the predictive performance is usually poorer than other, more sophisticated, algorithms. The performance of the present linear regression model is illustrated in Figure 9, showing scatterplots plotting the predicted versus the actual values for flight price (with log transformation). It looks like the linear regression model performs better on the LCC than the

FSC dataset, since the majority of the data points are situated closer to the regression line than for the FSC data. Overall, the plots reveal that the general pattern is captured by the models, yet, there are still numerous predictions that deviate significantly from the true values.

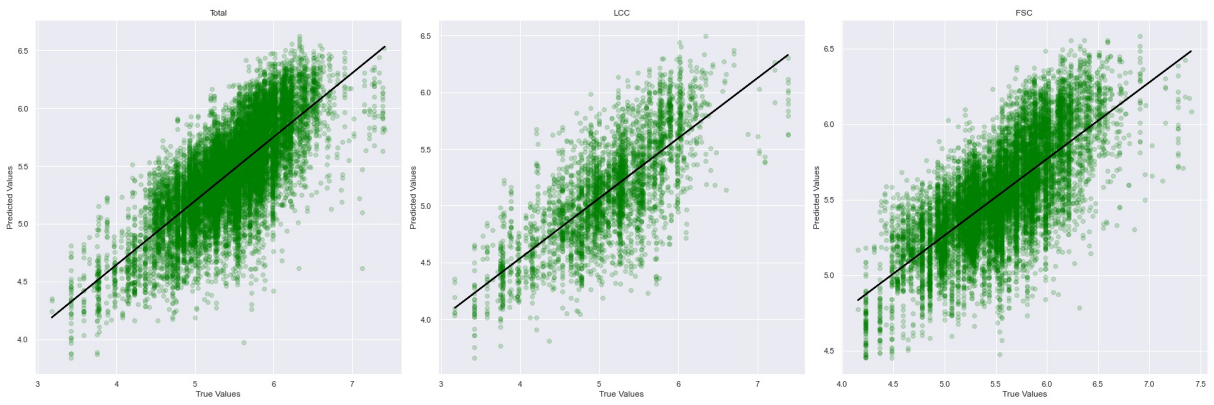
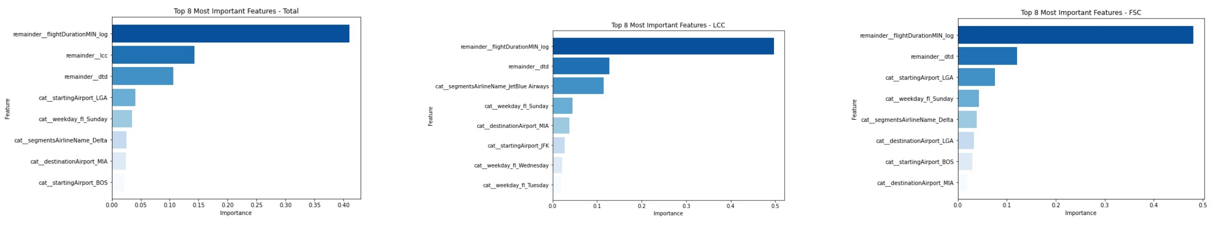


Figure 9: Linear Regression: True vs predicted values per dataset

With regard to the error metrics, the R^2 values range between 0.51 and 0.57 indicating moderate predicting power. The highest value is observed for the total dataset, whereas the model performed poorest on the FSC dataset. This is in line with what Figure 9 revealed. For all three variations of the linear regression model the computational time is very low.

Next, the results of the KNN models show an overall good performance. The R^2 values lie between 0.67 and 0.72. Notably, the computational time is extremely short for the LCC dataset (0.70 seconds), whereas it is 6.85 seconds for the FSC data and over the double for the total dataset (14.12 seconds). The set default value for K is 5. The model was additionally tested on all three datasets with $K = 3$ and $K = 7$, however, the R^2 values decreased in both scenarios, resulting in $K = 5$ for all the KNN models used in this thesis.

The Gradient Boosting models show slightly poorer results than the KNN model described before. Additionally, the computational time is significantly higher. After deploying the Gradient Boosting Regressor for all the datasets, their respective feature importance was plotted, visualizing how influential each of the features are. Feature importance gives an insight into how valuable each of the predictors is relative to the others. The higher the importance, the more influential the feature towards to output variable. Figure 9 shows the eight most prominent features that the algorithm detected in each dataset. Comparing the outcomes, it becomes obvious that flight duration is the key driver for flight price for LCC, FSC, as well as the combined dataset. After that, days to departure play the second most important role for both the LCC and the FSC data. For the combined dataset, days to departure ranks as the third most significant predictor, while the classification between LCC and non-LCC holds the second place in terms of importance. Based on these plots, it can be understood that the most influential factors for the flight price seem to be relatively similar across the datasets. Moreover, it is worth noting that Sunday as flight days is within the top 5 features in all the datasets. As was illustrated in Figure 1 in section 4 of this thesis, Sunday seems to be the most expensive day to fly. However, when looking at the features ranked below the most important ones, larger discrepancies become apparent, listing different airlines and airports as significant characteristics.



(a) Total (b) LCC (c) FSC

Figure 10: Feature importance per dataset

The Decision Tree models, on the other hand, have similar results to the GB models but way lower computational time. Figure 11 shows the first three layers of the model's output for the total dataset. The root node, situated at the top, represents the starting point of the data splitting process. One can see that flight duration again is the most important feature, determining the initial data split.

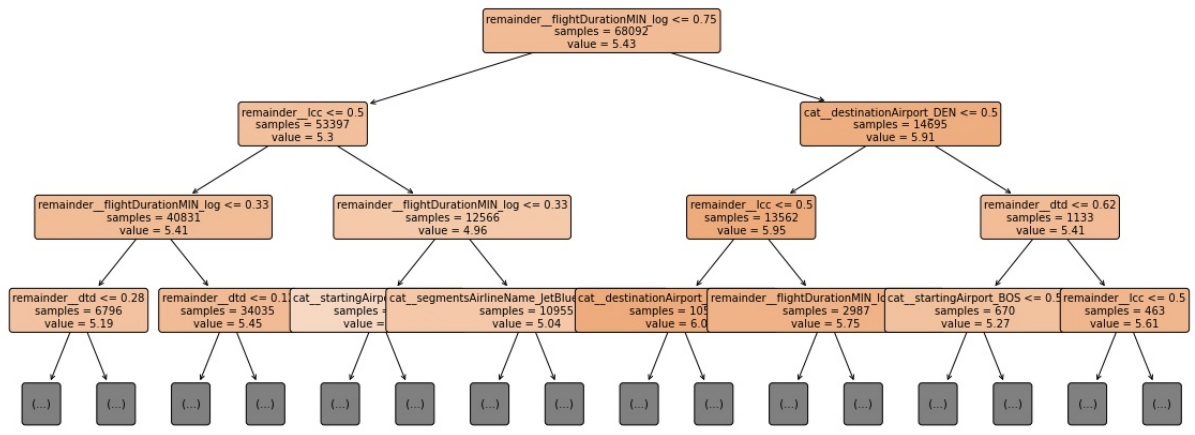


Figure 11: Decision tree - total dataset

-

Lastly, the Random Forest models outperform all the previous models materially. They show the highest R^2 for all three datasets, ranging from 0.81 for the FSC dataset to 0.83 for the total dataset. Yet, the computational time is also remarkably higher for these algorithms.

Overall, the computational time for the LCC dataset is always the shortest. Similarly, the error rates are consistently higher for the LCC dataset. This might be due to the fact that the LCC dataset has considerably less observations than the other two datasets and therefore the algorithms have fewer data points from which they can learn. This might be the reason for poorer results. Additionally, as demonstrated in Figure 7 in section 4.2, the LCC dataset shows higher volatility, which could also explain the higher error rates.

All in all, the Random Forest models show the best results but are also computationally the most expensive. The decision trees and KNN algorithms have a slightly lower performance but are computationally more efficient. Finding a balance between predictive performance and computational efficiency is a trade-off that is always present when evaluating different machine learning models. Which model is the best fit depends on the computational resources available.

The complete code in Python can be found in Appendix A.4.

6 Discussion

This chapter includes the interpretation of the obtained results throughout this study and aims to answer the research questions. Moreover, the limitations are discussed.

6.1 Findings

The goal of this thesis was find out to what extent flight details are sufficient to predict flight prices, what the most influential determinants of ticket prices are and whether they differ between low-cost and legacy carriers. Ultimately, the aim was to identify which machine learning algorithm could best forecast flight fares.

In terms of the sufficiency of flight details as input for price forecasting, the analysis reveals that a big part of the price is determined by flight details such as days to departure and flight duration. At the same time, this thesis' best performing model, the Random Forest of the total dataset, reached a R^2-value of 0.83. While this is a fairly decent result, it still leaves almost 20% of the variability in the dataset unexplained. This indicates that other external factors also influence ticket prices. These might be fuel prices, travel restrictions and geopolitical events, to name a few.

With regard to the most influential determinants of flight prices flight duration is, as expected, the most important feature. The longer the flight distance, the higher the price. This is true for LCC and FSC. Following flight duration, airline type plays the second most important role for the mixed dataset. LCC do show lower prices than FSC almost throughout the entire analysis. The only time average LCC prices exceed those of FSC is when 0-3 seats are left in a specific flight.

For both the LCC and the FSC dataset days to departure are the next most influential features after flight duration. By examining the relationship between average total fare and days to departure it became evident that the prices increase considerably around 9 days to departure and peak at 4 days remaining, after which they decrease steadily until the departure day. This is most probably because the airlines want to sell the remaining seats so they lower the prices (Groves and Gini, 2011).

With respect to seats remaining and their impact towards the flight price, it was found that their influence is not as clear as was expected. The connection between the two variables was examined in section 4, which already gave an idea of their importance in the models created subsequently. The plot (Figure 7) showed that the average price for FSC hardly changes when the number of seats remaining decrease. For LCC the prices fluctuate more depending on the seats remaining. While they start comparable low at 10 seats remaining they increase considerably, with an exception at four days remaining, and reach the highest values when only one seat is left. This contradicts the finding of the study conducted by Etzioni et al. (Etzioni et al., 2003), namely that LCC fluctuate less than FSC.

Similarly, the models did not identify the number of seats remaining as very influential factor. The feature importance from the Gradient Boosting Regressor lists seats remaining on the 12^{th} place for the combined dataset and FSC separately, and only on 24^{th} place for LCC. The Decision Tree also only shows the seats remaining on the 5^{th} depth level. This finding indicates that the number of seats remaining only becomes relevant further down in the feature selection.

Relating to flight day the analysis reveals that both LCC and FSC have a very similar price movement with relation to the day of the week. In both datasets, Sunday turns out to be the most expensive flight day, whereas the flight fares fall at the beginning of the week, reaching the lowest point on Wednesday. After that, the prices increase towards the weekend, decrease a little on Saturday and rise sharply on Sunday. This observations aligns with the findings of Malighetti et al. (Malighetti et al., 2015) who also concluded that prices tend to peak around Sunday and are cheapest on Wednesday and Thursday.

As for the performance of machine learning models to forecast flight prices various algorithms were tested throughout this thesis. Each was deployed three times, with the combined dataset, only for full service carriers, and lastly only for low cost carriers. While, as expected, the linear Regression performed the worst for all datasets, the Random Forest had the best predictive performance in all three cases ranging from a R^2 value from 0.81 for the FSC data to 0.82 and 0.83 for the LCC and the combined dataset, respectively. That being said, the superior performance comes with a higher computational cost as well. The Random Forest models all show longer computational times than any other models. This trade-off is a common consideration when deciding between different models. Whether a more complex model with higher computational cost or a simpler model with slightly poorer prediction accuracy is preferred depends on the specific problem and requirements at hand. In this case, despite the Random Forest having the longest running time, it still is the preferred choice since the duration is manageable especially considering the huge dataset used in this analysis. For this reason the Random Forest is selected as the best model to forecast flight prices for all the three scenarios, being LCC, FSC and both airline types combined, analysed throughout the thesis.

6.2 Limitations

Lastly, it is important to acknowledge the limitations and scope of this research to interpret the results appropriately and get an understanding of how well the findings can be generalized. Three main constraints were identified throughout this thesis.

Firstly, the maximum number of days to departure in the dataset is limited to 39. In reality, it is possible to book flights much further in advance already. Depending on the nature of the trip, such as business or leisure, tickets are booked earlier or later. Still, by only having a maximum of 39 days to departure, the price dynamics for tickets purchased well in advance might not be captured in this thesis.

Similarly, the range of the number of seats remaining in the dataset spans from zero to ten. Since the typical commercial airplane accommodates significantly passengers, having only 10 seats remaining indicates that the flight is nearly fully booked, hence, comparably higher prices are expected.

Furthermore, only one search date is considered in the present analysis, meaning the price development over time for individual flights could not be examined. Additionally, the dataset includes observations from the year 2022. Consequently, it can be assumed that the aviation industry has not yet fully recovered from the impact of the COVID-19 pandemic. As a results, the prices listed in this dataset might still be influenced by the repercussions of the travel restrictions the preceding years.

These identified constraints must be kept in mind when generalizing the results of this study while at the same time presenting opportunities for further investigations in this area.

7 Conclusion

To conclude, this thesis analysed that flight prices can be predicted pretty accurately using only flight details such as flight duration and days to departure. The best performing machine learning model is the Random Forest, which for each of the datasets, them being LCC, FSC and both combined, reached R^2 values of 0.81, 0.82 and 0.83. Even though these models also have the highest computational time, their higher predictive performance outweigh the higher computational cost. Nevertheless, it is important to mention that almost 20% of the variability in the data remains unexplained, meaning that other external factors are necessary to reach a higher accuracy.

Concerning the most important features, it was found that flight duration and days to departure play a significant role in determining the flight price. This is true for all three datasets and is in line with what previous literature covering this topic have pointed out. The number of seats remaining, on the other hand, seems to only influence the price on a deeper level, after having considered various features beforehand. This became evident during the feature importance analysis of the Gradient Boosting Regressor as well as the Random Forest. This finding was somewhat surprising since intuitively and according to other studies this factor plays an important role in price determination. However, this result might be attributed to the fact that the dataset only had up to 10 seats remaining, thus, limited variation and therefore the feature's potential relevance may not have been effectively demonstrated.

As mentioned previously, certain limitations of this study need to be taken into careful consideration when generalizing the results. At the same time, these constraints present opportunities for further research in this field. First of all, extending the dataset to include more than ten seats remaining as well as over 39 days to departure would allow to study the price developments over a longer period of time and get an insight into the price changes from when the plane is still empty to when it is almost fully booked. Apart from including longer time horizons in the analysis, it would make sense to include a time series component, by investigating the price developments of specific flights to understand the individual pricing patterns and developments of the flight. This would effectively allow researchers to get a more detailed insight into a specific price evolution. Furthermore, the present dataset only includes national flights within in the USA. A possible next step would be to enlarge the analysis for international flights. Also, the approach of using machine learning models to forecast flight prices could be enhanced by experimenting with deep learning algorithms that might convince with a higher predictive accuracy.

Lastly, instead of forecasting flight prices, one could build on existing literature aiming to supply a *buy* or *wait* decision, depending whether the prices are expected to increase or decrease after the search date.

References

Boer, A. V. D. (2015, 6). Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science 20*, 1–18.

Breiman, L. (2001). Random forests. *Machine Learning 45*, 5–32.

Coto-Millán, P., X. L. Fernández, P. Casares-Hontañón, V. Inglada, and M. Ángel Pesquera (2015). Assessing two airline models: Legacy vs. low cost carriers. *International Journal of Transport Economics 42*, 487–506.

Donaghy, K., U. Mcmahon, and D. Mcdowell (1995). Yield management: an overview. *International journal of hospitality management 14*, 139–150.

Etzioni, O., C. A. Knoblock, R. Tuchinda, and A. Yates (2003). To buy or not to buy: Mining airfare data to minimize ticket purchase price. pp. 119–128. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.

Friedman, J. H. (2006, 9). Recent advances in predictive (machine) learning. *Journal of Classification 23*, 175–197.

Gordiievych, A. and I. Shubin (2015, 10). Forecasting of airfare prices using time series. pp. 68–71. IEEE.

Groves, W. and M. Gini (2011). A regression model for predicting optimal purchase timing for airline tickets.

Groves, W. and M. Gini (2013). An agent for optimizing airline ticket purchasing. pp. 1341–1342.

Groves, W. and M. Gini (2015, 9). On optimizing airline ticket purchase timing. *ACM Transactions on Intelligent Systems and Technology 7*, 1–28.

James, G., D. Witten, T. Hastie, and R. Tibshirani (2021). *An Introduction to Statistical Learning with Applications in R Second Edition* (2 ed.). Springer Publication.

Kimes, S. E. (1989). The basics of yield management. *Cornell Hotel and Restaurant Administration Quarterly 30*, 14–19.

Kwoka, J., K. Hearle, and P. Alepin (2016, 5). From the fringe to the forefront: Low cost carriers and airline price determination. *Review of Industrial Organization 48*, 247–268.

Lantseva, A., K. Mukhina, A. Nikishova, S. Ivanov, and K. Knyazkov (2015). Data-driven modeling of airlines pricing. Volume 66, pp. 267–276.

Malighetti, P., S. Paleari, and R. Redondi (2015, 9). Easyjet pricing strategy: determinants and developments. *Transportmetrica A: Transport Science 11*, 686–701.

Papadakis, M. (2012). Predicting airfare prices.

Provost, F. and T. Fawcett (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking.* O'Reilly Media, Inc.

Rajankar, S. and N. Sakharkar (2019). A survey on flight pricing prediction using machine learning. *Internatıonal Journal Of Engıneerıng Research Technology (Ijert) 8*, 1281–1284.

Rajure, P., S. Bankar, H. Bakshi, and B. Patil (2021). Prediction of domestic airline tickets using machine learning. *International Journal for Research in Applied Science and Engineering Technology 9*, 666–674.

Rozenberg, R., S. Szabo, and I. Šebeščáková (2014). Comparison of fsc and lcc and their market share in aviation. *International Review of Aerospace Engeneering (IREASE) 7*, 149–154.

Selcuk, A. M. and Z. M. Avşar (2019, 10). Dynamic pricing in airline revenue management. *Journal of Mathematical Analysis and Applications 478*, 1191–1217.

Taddy, M. (2019). *Business data science: Combining machine learning and economics to optimize, automate, and accelerate business decisions.* McGraw-Hill Education.

Tsoukalas, G., P. Belobaba, and W. Swelbar (2008). Cost convergence in the us airline industry: An analysis of unit costs 1995–2006. *Journal of Air Transport Manegement 14*, 179–187.

Varian, H. R. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives 28*, 3–28.

Wang, T., S. Pouyanfar, H. Tian, Y. Toa, M. Alonso, S. Luis, and S.-C. Chen (2019). A framework for airfare price prediction: a machine learning approach. pp. 200–207.

Wang, Z. (2020). Rwa: A regression-based scheme for flight price prediction.

A Appendix

A.1 Raw dataset

Variable	Description
legID	An identifier for the flight
searchDate	The date (YYYY-MM-DD) on which this entry was taken from Expedia
flightDate	The date (YYYY-MM-DD) of the flight
startingAirport	Three-character IATA airport code for the initial location
destinationAirport	Three-character IATA airport code for the arrival location
fareBasisCode	The fare basis code
travelDuration	The travel duration in hours and minutes
elapsedDays	The number of elapsed days
isBasicEconomy	Boolean for whether the ticket is for basic economy
isRefundable	Boolean for whether the ticket is refundable
isNonStop	Boolean for whether the flight is non-stop
baseFare	The price of the ticket (in USD)
totalFare	The price of the ticket (in USD) including taxes and other fees
seatsRemaining	Integer for the number of seats remaining
totalTravelDistance	The total travel distance in miles. This data is sometimes missing
segmentsDepartureTimeEpochSeconds	String containing the departure time (Unix time) for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsDepartureTimeRaw	String containing the departure time (ISO 8601 format: YYYY-MM-DDThh:mm:ss. 000±[hh]:00) for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsArrivalTimeEpochSeconds	String containing the arrival time (Unix time) for each leg of the trip. The entries for each of the legs are separated by '||'

Variable	Description
segmentsArrivalTimeRaw	String containing the arrival time (ISO 8601 format: YYYY-MM-DDThh:mm:ss.000±[hh]:00) for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsArrivalAirportCode	String containing the IATA airport code for the arrival location for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsDepartureAirportCode	String containing the IATA airport code for the departure location for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsAirlineName	String containing the name of the airline that services each leg of the trip. The entries for each of the legs are separated by '||'
segmentsAirlineCode	String containing the two-letter airline code that services each leg of the trip. The entries for each of the legs are separated by '||'
segmentsEquipmentDescription	String containing the type of airplane used for each leg of the trip (e.g. "Airbus A321" or "Boeing 737-800"). The entries for each of the legs are separated by '||'
segmentsDurationInSeconds	String containing the duration of the flight (in seconds) for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsDistance	String containing the distance traveled (in miles) for each leg of the trip. The entries for each of the legs are separated by '||'
segmentsCabinCode	String containing the cabin for each leg of the trip (e.g. "coach"). The entries for each of the legs are separated by '||'

A.2 Data pre-processing in R

```r
#install.packages("corrplot")
#install.packages("car")
#install.packages("Metrics")
#install.packages("GGally")



#load packages
library(dplyr)
library(data.table)
library(ggplot2)
library(pwr)
library(broom)
library(lubridate)
library(scales)
library(tidyverse)
library(ggfortify)
library(powerMediation)
library(skimr)
library(stargazer)
library(lmtest)
library(readr)
library(vip)
library(tidymodels)
library(themis)
library(corrplot)
library(car)
library(Metrics)
library(GGally)



#empty environment
rm(list = ls())

#check files in wd
getwd()
setwd("C:/Users/sophi/OneDrive/Dokumente/Catolica/Master_thesis/Data")
list.files()
```

```
#load data
#dt <- read.csv("itin_2M.csv")
#table(dt$searchDate)

#data pre-processing
#filter for only 2022-02-22 - Wednesday
data <- dt %>% filter(searchDate == "2022-04-20")

#only non-stop flights
data_nonstop_raw <- data %>% filter(isNonStop == 1)
view(data_nonstop)

#check distinct IDs to make sure every flight appears only once
#n_distinct(data_nonstop$legId)

#remove other datasets
rm(dt)
rm(data)

#create csv with raw non-stop data for April 22 2022
#write.csv(data_nonstop_raw, "data_nonstop_raw_20220422.csv")
data_nonstop_raw <- read.csv("data_nonstop_raw_20220422.csv")

#check datatypes
str(data_nonstop_raw)

#change datatypes
data_nonstop <- data_nonstop_raw %>%
  mutate(isBasicEconomy = as.numeric(isBasicEconomy),
         isRefundable = as.numeric(isRefundable),
         flightDate = as.Date(flightDate),
         searchDate = as.Date(searchDate),
         segmentsDurationInSeconds =
           as.numeric(segmentsDurationInSeconds))

#check datatypes after casting
#sapply(data_nonstop[, c("flightDate","searchDate",
#                        "isRefundable")], class)
```

```
#str(data_nonstop)

#create variable days to departure
#convert seconds into minutes
data_nonstop <- data_nonstop %>% mutate(
  dtd = as.numeric(flightDate - searchDate),
  flightDurationMIN = segmentsDurationInSeconds/60)

#extract departure time
data_nonstop <- data_nonstop %>%
  mutate(dep_time = substr(segmentsDepartureTimeRaw, 12, 16))

#split time of day in 4 categories
data_nonstop <- data_nonstop %>%
  mutate(day_time = case_when(
    between(hour(as.POSIXlt(dep_time, format = "%H:%M")), 0, 5) ~
      "Night",
    between(hour(as.POSIXlt(dep_time, format = "%H:%M")), 6, 11) ~
      "Morning",
    between(hour(as.POSIXlt(dep_time, format = "%H:%M")), 12, 17) ~
      "Afternoon",
    between(hour(as.POSIXlt(dep_time, format = "%H:%M")), 18, 23) ~
      "Evening"
  ))

#filter for only coach
#table(data_nonstop_raw$segmentsCabinCode)
data_nonstop <- data_nonstop %>% filter(segmentsCabinCode == "coach")

#isbasiceconomy
table(data_nonstop_raw$isBasicEconomy)
data_nonstop <- data_nonstop %>% filter(isBasicEconomy == '0')

#relationship between weekday and price
#add the weekdays for the flight dates
data_nonstop <- data_nonstop %>% mutate(weekday_fl = weekdays(flightDate))

# Perform ANOVA test bec multiple groups (7 days)
weekday_price <- aov(totalFare ~ weekday_fl, data = data_nonstop)
```

```
weekday_price_result <- summary(weekday_price)
print(weekday_price_result)

#Tukey's test for pairwise comparison of means
weekday_price_tukey <- TukeyHSD(weekday_price)
print(weekday_price_tukey)

#relationship between airline name and price
# Perform ANOVA test bec multiple groups (7 airline names)
table(data_nonstop$segmentsAirlineName)

airline_price <- aov(totalFare ~ segmentsAirlineCode, data = data_nonstop)
airline_price_result <- summary(airline_price)
print(airline_price_result)

#result: significant difference between groups bec very small p-value

#Tukey's test for pairwise comparison of means
airline_price_tukey <- TukeyHSD(airline_price)
print(airline_price_tukey)

#create dummy for low cost airlines
#table(data_nonstop$segmentsAirlineCode)
data_nonstop <- data_nonstop %>% mutate(
  lcc = ifelse(segmentsAirlineCode == 'B6' | segmentsAirlineCode == 'F9'|
                    segmentsAirlineCode == 'NK', 1, 0))
table(data_nonstop$lca)

#diff between lcc and legacy carriers
ttest_lcc <- t.test(price_min ~ lcc, data = data_nonstop)
print(ttest_lcc)

#all values in isRefundable are 0 so that column can be removed
unique(data_nonstop_raw$isRefundable)

#remove elapseddays column bec travelduration and dep time sufficient
table(data_nonstop_raw$elapsedDays)
```

```
#delete unnecessary columns
data_nonstop <- data_nonstop %>% select(-X, -X.1, -legId, -searchDate,
                                        -isNonStop, -fareBasisCode,
                                        -isRefundable, -travelDuration,
                                        -elapsedDays,
                                        -segmentsDurationInSeconds,
                                        -baseFare, -segmentsDistance,
                                        -segmentsCabinCode,
                                        -segmentsAirlineCode,
                                        -segmentsEquipmentDescription,
                                        -segmentsArrivalAirportCode,
                                        -segmentsDepartureAirportCode,
                                        -isBasicEconomy,
                                        -segmentsArrivalTimeEpochSeconds,
                                        -segmentsArrivalTimeRaw,
                                        -segmentsDepartureTimeEpochSeconds,
                                        -segmentsDepartureTimeRaw,
                                        -flightDate)

#create csv
write.csv(data_nonstop, "data_nonstop_preprocessed.csv")
```

A.3 Pre-processed dataset

Variable	Description
startingAirport	String containing the IATA airport code for the departure location
destinationAirport	String containing the IATA airport code for the arrival location
totalFare	The price of the ticket (in USD) including taxes and other fees
seatsRemaining	The number of seats remaining
totalTravelDistance	The total travel distance in miles
segmentsAirlineName	String containing the name of the airline that services the trip
dtd	The number of days to departure calculated by the difference between search date and flight date
flightDurationMIN	The flight duration in minutes calculated by dividing the variable segmentsDurationInSeconds by 60
dep_time	String containing the departure time of the flight (format: hh:mm) extracted from the variable segmentsDepartureTimeRaw
day_time	String containing whether the flight is in the morning, afternoon, evening or during the night
weekday_fl	String containing the day of the week on which the flight departs
lcc	Dummy for whether the airline is a low-cost-carrier or not

A.4 Exploratory Data Analysis in Python

```
### !/usr/bin/env python3
# -*- coding: utf-8 -*-
# imports
import os as os, numpy as np, pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import statsmodels.api as sm
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn import svm
from sklearn import tree
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import learning_curve
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error, r2_score,
    mean_absolute_error
import warnings
import seaborn as sns
import time
warnings.filterwarnings("ignore")

#import data
import_data_raw = pd.read_csv('../Data/data_nonstop_preprocessed.csv',
    delimiter=',')
import_data_raw.shape
import_data_raw.dtypes

Numerical Variables
import_data.describe()
```

```
#Price per weekday
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
    'Friday', 'Saturday', 'Sunday']

# Price per weekday for low-cost airlines
average_price_lcc = import_data[import_data['lcc'] == 1].groupby
    ('weekday_fl')['totalFare'].mean()
average_price_lcc = average_price_lcc.reindex(weekday_order)

# Price per weekday for legacy airlines
average_price_legacy = import_data[import_data['lcc'] == 0].groupby
    ('weekday_fl')['totalFare'].mean()
average_price_legacy = average_price_legacy.reindex(weekday_order)

# Plot the average price per weekday for both airline type
price_weekday = plt.figure(figsize=(9, 4))
sns.lineplot(data=average_price_lcc, color='blue',
    label='Low-Cost␣Airlines')
sns.lineplot(data=average_price_legacy, color='red',
    label='Legacy␣Airlines')
plt.xlabel('Weekday')
plt.ylabel('Average␣Price␣(USD)')
#plt.title('Average Price Per Flight Day')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()
#price_weekday.savefig('price_weekday.jpg', format='jpg',
    bbox_inches='tight')



# Distribution of total fare with and without log transformation
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot without log transformation
sns.histplot(import_data['totalFare'], kde=True, ax=axes[0])
axes[0].set_xlabel('Total␣Fare')
axes[0].set_ylabel('Frequency')
```

```
axes[0].set_title('Distribution␣of␣Total␣Fare
␣␣␣␣(Without␣Log␣Transformation)')

# Plot with log transformation
sns.histplot(np.log1p(import_data['totalFare']), kde=True, ax=axes[1])
axes[1].set_xlabel('Log␣Transformed␣Total␣Fare')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Distribution␣of␣Total␣Fare
␣␣␣␣(With␣Log␣Transformation)')

# Adjust spacing between subplots
plt.tight_layout()
# Show the plot
plt.show()
#fig.savefig('totalfare_log.jpg', format='jpg')

#price distribution per airline
fig_total = plt.figure(figsize=(16, 8))
sns.set(style="darkgrid")
sns.boxplot(x='segmentsAirlineName', y='totalFare', data=import_data)
plt.xlabel('Airline')
plt.ylabel('Total␣Flight␣Price')
plt.title('Price␣Distribution␣per␣Airline')
plt.xticks(rotation=45)
plt.show()
#fig_total.savefig('airlines_total.jpg', format='jpg')

#log transformation of totalfare and flightduration
import_data[['totalFare_log',
    'flightDurationMIN_log']] = import_data[['totalFare',
    'flightDurationMIN']].apply(np.log)

#price distribution per airline with log
fig_log = plt.figure(figsize=(16, 8))
sns.set(style="darkgrid")
sns.boxplot(x='segmentsAirlineName', y='totalFare_log',
    data=import_data)
plt.xlabel('Airline')
plt.ylabel('Total␣Flight␣Price')
plt.title('Price Distribution per Airline after log transformation')
plt.xticks(rotation=45)
plt.show()
#fig_log.savefig('airlines_log.jpg', format='jpg')

#corr matrices
data_lcc_1 = import_data[import_data['lcc'] == 1].drop(columns=['lcc'])
data_lcc_0 = import_data[import_data['lcc'] == 0].drop(columns=['lcc'])
corr_matrix_lcc_1 = data_lcc_1.corr()
corr_matrix_lcc_0 = data_lcc_0.corr()
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))
sns.heatmap(corr_matrix_lcc_1, annot=True, cmap='coolwarm',
    square=True, ax=ax1)
sns.heatmap(corr_matrix_lcc_0, annot=True, cmap='coolwarm',
    square=True, ax=ax2)
ax1.set_title('Low Cost Carrier')
ax2.set_title('Full Service Carrier')
plt.tight_layout()
plt.show()
fig.savefig('corr_matrices.jpg', format='jpg')

#columns with NAs
cols_with_nas = import_data.columns[import_data.isna().any()].tolist()
print(cols_with_nas)

# Average Total Fare by Days to Departure
grouped_data = import_data.groupby(['lcc', 'dtd'])
    ['totalFare'].mean().reset_index()
#sort unique dtd value in desceding order
sorted_dtd = sorted(grouped_data['dtd'].unique(), reverse = True)
fig_dtd = plt.figure(figsize=(16, 8))
sns.barplot(data=grouped_data, x='dtd', y='totalFare', hue='lcc',
    order = sorted_dtd)
# Set the x-axis and y-axis labels
plt.xlabel('Days to Departure')
plt.ylabel('Average Total Fare')
plt.title('Average Total Fare by Days to Departure')
plt.legend(title='LCC')
plt.show()
```

```
fig_dtd.savefig('avgfare_dtd.jpg', format='jpg')

# Average Total Fare by Number of Seats Remaining
grouped_data = import_data.groupby(['lcc', 'seatsRemaining'])
    ['totalFare'].mean().reset_index()
# Sort the unique values of dtd in descending order
sorted_seats = sorted(grouped_data['seatsRemaining'].unique(),
    reverse=True)
fig2 = plt.figure(figsize=(16, 8))
sns.barplot(data=grouped_data, x='seatsRemaining', y='totalFare',
    hue='lcc', order=sorted_seats)
# Set the x-axis and y-axis labels
plt.xlabel('Number␣of␣Seats␣Remaining')
plt.ylabel('Average␣Total␣Fare')
plt.title('Average␣Total␣Fare␣by␣Number␣of␣Seats␣Remaining')
plt.legend(title='LCC')
plt.show()
#fig2.savefig('avgfare_seats.jpg', format='jpg')>


Numerical Variables
import_data.describe(include='O')

# Distribution of airlines
overall_distribution = import_data['segmentsAirlineName'].value_counts()
# Group the data by lcc and calculate the count for each group
grouped_data = import_data['lcc'].value_counts()
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
axes[0].pie(overall_distribution, labels=overall_distribution.index,
    autopct='%1.1f%%')
axes[0].set_title('Distribution␣of␣Airlines')
axes[1].pie(grouped_data, labels=grouped_data.index, autopct='%1.1f%%')
axes[1].set_title('Distribution␣of␣Airline␣type')
axes[1].legend(labels=['FSC', 'LCC'], loc='upper␣right')
plt.tight_layout()
plt.show()
fig.savefig('airlines_distr.jpg', format='jpg')
```

A.5 Modelling in Python

```
Normalising
num_cols = ['totalFare', 'seatsRemaining', 'dtd', 'flightDurationMIN']
import_data_raw[num_cols].hist();
import_data_raw[num_cols].hist(figsize=(10,11));

#Distribution of fllight duration with and without log transformation
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
# Plot without log transformation
sns.histplot(import_data['flightDurationMIN'], kde=True, ax=axes[0])
axes[0].set_xlabel('Flight␣Duration')
axes[0].set_ylabel('Frequency')
axes[0].set_title('Distribution␣of␣Flight␣Duration
␣␣␣␣(Without␣Log␣Transformation)')
# Plot with log transformation
sns.histplot(np.log1p(import_data['flightDurationMIN_log']),
    kde=True, ax=axes[1])
axes[1].set_xlabel('Log␣Transformed␣Flight␣Duration')
axes[1].set_ylabel('Frequency')
axes[1].set_title('Distribution␣of␣Flight␣Duration
␣␣␣␣(With␣Log␣Transformation)')
plt.tight_layout()
plt.show()
#fig.savefig('flightdur_log.jpg', format='jpg')

Scaling
from sklearn.preprocessing import MinMaxScaler
# Create MinMaxScaler
scaler = MinMaxScaler()
#choose columns
scale_cols = ['dtd', 'seatsRemaining', 'flightDurationMIN_log']
import_data[scale_cols] = scaler.fit_transform(import_data[scale_cols])
#checke whether vaariables were scaled
import_data.describe()

Modelling
#create serparate dataset for lcc and fsc
data_lcc = import_data[import_data['lcc'] == 1]
```

```
data_fsc = import_data[import_data['lcc'] == 0]

Split in train and test data
#total dataset
X_train_total, X_test_total, y_train_total,
    y_test_total = train_test_split(
    import_data[['weekday_fl', 'seatsRemaining', 'dtd', 'lcc',
    'segmentsAirlineName', 'startingAirport',
    'destinationAirport', 'day_time',
    'flightDurationMIN_log']], import_data[['totalFare_log']],
    test_size=0.2, random_state=42
)

# lcc
X_train_lcc, X_test_lcc, y_train_lcc, y_test_lcc = train_test_split(
    data_lcc[['weekday_fl', 'seatsRemaining', 'dtd',
    'segmentsAirlineName', 'startingAirport',
    'destinationAirport', 'day_time',
    'flightDurationMIN_log']], data_lcc[['totalFare_log']],
    test_size=0.2, random_state=42)

# fsc
X_train_fsc, X_test_fsc, y_train_fsc, y_test_fsc = train_test_split(
    data_fsc[['weekday_fl', 'seatsRemaining', 'dtd',
    'segmentsAirlineName', 'startingAirport',
    'destinationAirport', 'day_time',
    'flightDurationMIN_log']], data_fsc[['totalFare_log']],
    test_size=0.2, random_state=42)

One-hot encode categorical variables
categorical_cols = ['weekday_fl', 'segmentsAirlineName', 'day_time',
    'startingAirport', 'destinationAirport']

def onehotencode_data(train_data, test_data, categorical_cols):
    categorical_encoders = OneHotEncoder(sparse=False)
    preprocessor = ColumnTransformer(transformers=[('cat',
    categorical_encoders, categorical_cols)], remainder='passthrough')
    # Preprocess the training data
    X_train_preprocessed = preprocessor.fit_transform(train_data)
    # Preprocess the test data
    X_test_preprocessed = preprocessor.transform(test_data)
    # Get the feature names
    feature_names = preprocessor.get_feature_names_out()
    return X_train_preprocessed, X_test_preprocessed, feature_names

# Preprocess the total dataset
X_train_total_preprocessed, X_test_total_preprocessed,
    feature_names_total = onehotencode_data(X_train_total,
    X_test_total, categorical_cols)

# Preprocess lcc dataset
X_train_lcc_preprocessed, X_test_lcc_preprocessed, feature_names_lcc =
    onehotencode_data(X_train_lcc, X_test_lcc, categorical_cols)

# Preprocess fsc dataset
X_train_fsc_preprocessed, X_test_fsc_preprocessed, feature_names_fsc =
    onehotencode_data(X_train_fsc, X_test_fsc, categorical_cols)

# Define the datasets and their corresponding variables for the models
datasets = {
    'Total': {
        'X_train': X_train_total_preprocessed,
        'X_test': X_test_total_preprocessed,
        'y_train': y_train_total,
        'y_test': y_test_total
    },
    'LCC': {
        'X_train': X_train_lcc_preprocessed,
        'X_test': X_test_lcc_preprocessed,
        'y_train': y_train_lcc,
        'y_test': y_test_lcc
    },
    'FSC': {
        'X_train': X_train_fsc_preprocessed,
        'X_test': X_test_fsc_preprocessed,
        'y_train': y_train_fsc,
        'y_test': y_test_fsc
    }
}

Linear Regression
# Create an empty dictionary to store the results
results_lr = {}

# Train and evaluate the linear regression model for each dataset
for dataset_name, dataset in datasets.items():
    model_lr = LinearRegression()
    # Start the timer
    start_time_lr = time.time()
    # Train the model
    model_lr.fit(dataset['X_train'], dataset['y_train'])
    # Predict the target variable for the test data
    y_pred_lr = model_lr.predict(dataset['X_test'])
    # Stop the timer
    end_time_lr = time.time()
    # Calculate the total time
    total_time_lr = round((end_time_lr - start_time_lr), 5)
    # Evaluate the model
    mse = mean_squared_error(dataset['y_test'], y_pred_lr)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(dataset['y_test'], y_pred_lr)
    r2 = r2_score(dataset['y_test'], y_pred_lr)
    # Store the results
    results_lr[dataset_name] = {
        'Model': model_lr,
        'MSE': mse,
        'MAE': mae,
        'RMSE': rmse,
        'R-squared': r2,
        'Computational␣Time': total_time_lr
    }

# Print the results
for dataset_name, result in results_lr.items():
    print(f"Results␣for␣{dataset_name}␣dataset:")
    print("MSE:", result['MSE'])
    print("MAE:", result['MAE'])
    print("RMSE:", result['RMSE'])
    print("R-squared:", result['R-squared'])
    print("Computational␣Time:", result['Computational␣Time'],
        "seconds")
    print()

#scatterplot of residuals with regression line
fig, axes = plt.subplots(1, 3, figsize=(24, 8))
# Plot the scatter plot for each dataset in a separate subplot
for i, (dataset_name, result) in enumerate(results_lr.items()):
    ax = axes[i]
    y_true = datasets[dataset_name]['y_test']
    y_pred = result['Model'].predict(datasets[dataset_name]['X_test'])
    ax.scatter(y_true, y_pred, alpha=0.2, color='green')
    ax.set_xlabel("True␣Values")
    ax.set_ylabel("Predicted␣Values")
    ax.set_title(f"{dataset_name}")
    # Convert y_true and y_pred to 1D arrays
    y_true = y_true.values.ravel()
    y_pred = y_pred.ravel()
    # Add regression line
    slope, intercept = np.polyfit(y_true, y_pred, 1)
    ax.plot(y_true, slope * y_true + intercept, color='black',
        linewidth=2)
plt.tight_layout()
plt.show()
fig.savefig('pred_vs_true_lr.jpg', format='jpg', bbox_inches='tight')
```

KNN

```
# Create an empty dictionary to store the results
results_knn = {}

for dataset_name, dataset in datasets.items():
    model_knn = KNeighborsRegressor(n_neighbors = 5)
    start_time_knn = time.time()
    model_knn.fit(dataset['X_train'], dataset['y_train'])
    y_pred_knn = model_knn.predict(dataset['X_test'])
    end_time_knn = time.time()
    total_time_knn = round((end_time_knn - start_time_knn), 5)
    mse = mean_squared_error(dataset['y_test'], y_pred_knn)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(dataset['y_test'], y_pred_knn)
    r2 = r2_score(dataset['y_test'], y_pred_knn)
    results_knn[dataset_name] = {
        'Model': model_knn,
        'MSE': mse,
        'MAE': mae,
        'RMSE': rmse,
        'R-squared': r2,
        'Computational Time': total_time_knn
    }

for dataset_name, result in results_knn.items():
    print(f"Results for {dataset_name} dataset:")
    print("MSE:", result['MSE'])
    print("MAE:", result['MAE'])
    print("RMSE:", result['RMSE'])
    print("R-squared:", result['R-squared'])
    print("Computational Time:", result['Computational Time'],
        "seconds")
    print()

Gradient Boosting Model
# Create an empty dictionary to store the results
results_gb = {}
for dataset_name, dataset in datasets.items():
    model_gb = GradientBoostingRegressor()
    start_time_gb = time.time()
    model_gb.fit(dataset['X_train'], dataset['y_train'])
    y_pred_gb = model_gb.predict(dataset['X_test'])
    end_time_gb = time.time()
    total_time_gb = round((end_time_gb - start_time_gb), 5)
    mse = mean_squared_error(dataset['y_test'], y_pred_gb)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(dataset['y_test'], y_pred_gb)
    r2 = r2_score(dataset['y_test'], y_pred_gb)
    results_gb[dataset_name] = {
        'Model': model_gb,
            'MSE': mse,
            'MAE': mae,
            'RMSE': rmse,
            'R-squared': r2,
            'Computational␣Time': total_time_gb
        }

for dataset_name, result in results_gb.items():
    print(f"Results␣for␣{dataset_name}␣dataset:")
    print("MSE:", result['MSE'])
    print("MAE:", result['MAE'])
    print("RMSE:", result['RMSE'])
    print("R-squared:", result['R-squared'])
    print("Computational␣Time:", result['Computational␣Time'],
        "seconds")
    print()

model_total_gb = results_gb['Total']['Model']
model_lcc_gb = results_gb['LCC']['Model']
model_fsc_gb = results_gb['FSC']['Model']

#feature importances for each dataset
importances = model_total_gb.feature_importances_
indices = np.argsort(importances)[::-1]
top_features_indices = indices[:8]
# Define a colormap
colormap = plt.cm.get_cmap('Blues')
# Plot feature importances
feat_lcc = plt.figure(figsize=(8, 6))
bars = plt.barh(range(len(top_features_indices)),
    importances[top_features_indices][::-1])
# Set the color of each bar based on its importance value
for i, bar in enumerate(bars):
    bar.set_color(colormap(i / len(top_features_indices)))
plt.yticks(range(len(top_features_indices)), [feature_names_total[i]
    for i in top_features_indices][::-1])
plt.xlabel("Importance")
plt.ylabel("Feature")
plt.title("Top␣8␣Most␣Important␣Features␣-␣Total")
plt.show()
feat_total.savefig('feat_total.jpg', format='jpg', bbox_inches='tight')

Decision tree
# Create an empty dictionary to store the results
results_dt = {}

for dataset_name, dataset in datasets.items():
    model_dt = DecisionTreeRegressor()
    start_time_dt = time.time()
    model_dt.fit(dataset['X_train'], dataset['y_train'])
    y_pred_dt = model_dt.predict(dataset['X_test']
    end_time_dt = time.time()
    total_time_dt = round((end_time_dt - start_time_dt), 5)
    mse = mean_squared_error(dataset['y_test'], y_pred_dt)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(dataset['y_test'], y_pred_dt)
    r2 = r2_score(dataset['y_test'], y_pred_dt)
    results_dt[dataset_name] = {
        'Model': model_dt,
        'MSE': mse,
        'MAE': mae,
        'RMSE': rmse,
        'R-squared': r2,
        'Computational Time': total_time_dt
    }

for dataset_name, result in results_dt.items():
    print(f"Results for {dataset_name} dataset:")
    print("MSE:", result['MSE'])
    print("MAE:", result['MAE'])
    print("RMSE:", result['RMSE'])
    print("R-squared:", result['R-squared'])
    print("Computational Time:", result['Computational Time'],
        "seconds")
    print()

model_total_dt = results_dt['Total']['Model']
model_lcc_dt = results_dt['LCC']['Model']
```

```
model_fsc_dt = results_dt['FSC']['Model']

# Visualize the decision tree
feature_names = feature_names_total.tolist()
dt_total = plt.figure(figsize=(30, 8))
tree.plot_tree(model_total_dt, feature_names=feature_names,
    filled=True, fontsize=10, max_depth = 4, impurity=False,
    rounded=True, proportion=False, precision=2,
    node_ids=False, class_names=None)
plt.show()
#dt_total.savefig('dt_total.jpg', format='jpg', bbox_inches='tight')

Random forest
# Create an empty dictionary to store the results
results_rf = {}

for dataset_name, dataset in datasets.items():
    model_rf = RandomForestRegressor()
    start_time_rf = time.time()
    model_rf.fit(dataset['X_train'], dataset['y_train'])
    y_pred_rf = model_rf.predict(dataset['X_test'])
    end_time_rf = time.time()
    total_time_rf = round((end_time_rf - start_time_rf), 5)
    mse = mean_squared_error(dataset['y_test'], y_pred_rf)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(dataset['y_test'], y_pred_rf)
    r2 = r2_score(dataset['y_test'], y_pred_rf)
    results_rf[dataset_name] = {
        'Model': model_rf,
        'MSE': mse,
        'MAE': mae,
        'RMSE': rmse,
        'R-squared': r2,
        'Computational␣Time': total_time_rf
    }

for dataset_name, result in results_rf.items():
    print(f"Results␣for␣{dataset_name}␣dataset:")
    print("MSE:", result['MSE'])
```