MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Ali Mohammad Asad** Personal ID number: **472589**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Branch of study: **Data Science**

II. Master's thesis details

Master's thesis title in English:

Pricing and data: long-distance bus routes

Master's thesis title in Czech:

Cenová politika a data: dálkové autobusové linky

Guidelines:

Dynamic pricing is a well-established mechanism for balancing supply and demand in resource allocation problems. Although widely deployed in practice, the details on dynamic pricing algorithms are not publicly available in most cases. The goal of this thesis is to analyze sales data of these providers to get insights into their pricing strategies, customers response to dynamic pricing and overall effectiveness of the operations of these providers.
Assignment:
1) Familiarize yourself with the problem of pricing of transportation services. Survey current state of the art methods for analysis of spatio-temporal and pricing data.
2) Clean and prepare the bus datasets for analysis.
3) Perform descriptive analysis of the datasets.
4) Based on the descriptive analysis, propose new data features for the predictive model.
5) Define a predictive model using new data features. Propose evaluation metrics.
6) Evaluate proposed predictive model.

Bibliography / sources:

[1] Christopher M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), 2006
[2] Bivand, Roger S., Pebesma, Edzer, Gómez-Rubio, Virgilio, Applied Spatial Data Analysis with R, 2013
[3] Stuart Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, 2010
[4] Kara Kockelman, T. Donna Chen, Katie Larsen, Brice Nichols, The Economics of Transportation Systems: A Reference for Practitioners, 2013

Name and workplace of master's thesis supervisor:

Ing. Jan Mrkos, Artificial Intelligence Center, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **14.02.2019** Deadline for master's thesis submission: **22.05.2020**

Assignment valid until: **20.09.2020**

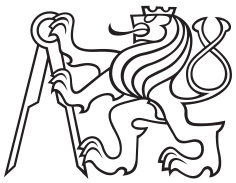
Ing. Jan Mrkos Supervisor's signature	Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Pricing and data: long-distance bus routes

Mohammad Asad Ali

Supervisor: Ing. Jan Mrkos
Field of study: Open Informatics
Subfield: Data Science
May 2020

Acknowledgements

I would like to express utmost of gratitude towards the supervisor for my thesis, Ing. Jan Mrkos for his immense patience and support in helping me deliver this thesis. From the inception of the topic to the very delivery of the final work, his guidance was always there when needed.

I would also take the opportunity to thank the teachers of subjects statistical data analysis and statistical machine learning, where the content of the subjects helped me towards learning the necessary skills required to deliver this work.

Last but not the least, a huge token of thanks to all my teachers for their contributions towards my educational development and Czech Technical University as a whole for giving me the chance of studying at this university and deliver this work.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources that have been used in accordance with the Guideline no. 1/2009 for adhering to ethical principles in the course of elaborating an academic final thesis.

Abstract

Bus transport is one of the most widely used means of transport around Europe. In this thesis the focus is on understanding the factors influencing pricing policies of long distance bus routes through exploratory analysis and use of machine learning techniques. To achieve this we have collected data from the two bus service providers and compared them against each other. Our results show the relationship between price and related features is non-linear and the major influential factors for both providers are hours to departure, distance and time traveled by the bus and the free capacity.

Keywords: pricing analysis, bus transportation, machine learning

Supervisor: Ing. Jan Mrkos
Praha 2, Karlovo náměstí 13, E-325

Abstrakt

Autobusová doprava je jeden z nejrozšířěnějších způsobů osobní dopravy v Evropě. V této práci se za použití technik strojového učení zaměřujeme na pochopení vlivů ovlivňujících dynamickou cenotvorbu na dálkových autobusových trasách. Za tímto účelem jsme sebírali data o cenách od dvou operátorů a porovnali je navzájem. Naše výsledky ukazují, že vztah mezi cenou a dalšími příznaky je nelineární. Hlavními faktory ovlivňujícími cenu u obou dopravců se ukázaly být příznaky *doba do odjezdu, délka cesty a volná kapacita autobusu.*

Klíčová slova: analýza cenotvorby, autobusová doprava, strojové učení

Překlad názvu: Cenová politika a data: dálkové autobusové linky

Contents

Figures

Tables

Chapter 1

Introduction

The aim of this thesis is to use machine learning techniques to analyse and understand the pricing strategies used by two of the leading bus travel agencies around Europe namely Student Agency and FlixBus.

Pricing of services and products is a very important factor for any business as it directly relates to its profitability. We seek to explore how two of the leading bus agencies across Europe are pricing their services, what are the major factors contributing towards it and how does it effect there overall sales. We would look to understand comprehensively, pricing mechanisms being employed by the bus agencies as to whether they are using static pricing i.e. the price of the service remains constant despite the change in demand or whether the more conventional and likely scenario of dynamic pricing i.e. the price of the service is changing with the change in demand is being employed by them. It could also be the case that the pricing methodology could turn out be a mix of both of the strategies mentioned previously and this is what we will discover through the course of the thesis.

Buses are one of the most widely used modes of transport both within and between the European countries. One of the main motivations behind the work is the opportunity to add value towards understanding the pricing dynamics as well as the passenger demand patterns. This could potentially help companies currently in the field to understand how their potential competitors are pricing their services and how their own weighs up to the competition. This work could also act as potential market research to new companies looking to enter the scene. Another group that could possibly benefit are the customers, through our prediction models we can suggest to them optimum ticket purchase dates as well as potential availability of seats in the bus for routes and dates they would like to travel.

Another motivational factor is the opportunity to contribute towards related literature on the subject matter. For example in [8], the authors' objectives are similar to ours. We can expand on their work further by presenting a comparative analysis of two bus agencies along with the application of some of the current state of the art machine learning techniques to enhance the analysis and bring a new contribution in the area.

As to what follows in the sections beyond the introduction, they can be divided into three parts - the first part of the thesis is dedicated to

presenting the overall task at hand, the motivation for doing it and the review of associated literature around the topic. The concluding sections introduce the theoretical background to the machine learning techniques which will be employed along with their evaluation metrics. The second part is centered primarily around introducing the data at hand. They describe data collection methodology, pre-processing and feature engineering applied and the comparative exploratory analysis for the two bus agencies. The third part is focused on training, testing and evaluation of our machine learning models followed by the their interpretation and discussion of results.

Chapter 2

Objectives of the thesis

The main focus of this thesis is to develop an understanding of the pricing strategies the used by the bus service providers. Our plan to achieve this can be divided into two parts. First part involves exploratory analysis of the datasets where we will try to understand the relationship between both the price of a ticket and free capacity of the bus with respect to all other features in the datasets. While the second part would revolve around training machine learning models on the datasets.

If we are able to train a good model(s) which are able to predict the price and the free capacity well, it would further allow us to drill down and see both on local and as well as a global level which features are exerting major influence over the price and free capacity of buses. What makes our model to be considered good will be decided based on the ability of model to generalize on the samples of the data it has not seen during training (referred to as testing set in the sections to come) by calculating evaluation metrics such as RMSE, MAE and Adjusted R-Squared .

Chapter 3

Machine Learning techniques

In this section we will be reviewing the techniques used in some of the related literature. We will compare their use-cases and data requirements to our use-case. Furthermore, we will present the theoretical background to the machine learning techniques which we think can work for our use-case based on the literature review and the nature of our datasets.

3.1 A quick review of methods used in existing literature

In this section we will shortly review some of the methods used in the some related literature and if they could be used for our case. While in essence the focus is same in [4] i.e. to forecast the demand but in our case the problem is much more complicated due to presence of more number of features along with a mix of numerical and categorical feature types with multiple levels. Moreoever, the datasets used in this thesis are very large (both datasets are having approximately 200k rows each).

Similarly in [12] and [8] much simpler machine learning techniques are being used and given the scale of our datasets combined with recent advancements in machine learning it does not make much sense to replicate them for our use case.

After a review of the existing work we observed that much of it has been carried out using old techniques and with the vast advancement in the field of machine learning plus considering the size of our datasets which is quite large it would make more sense to use advanced machine learning techniques such as ensembling.

3.2 Ensembling

Ensembling is a class of machine learning techniques based upon the idea of using aggregated output from multiple weaker models to give rise to a stronger model.

Before explaining ensembling further, it is important to understand a classical conundrum in statistics and machine learning which is the bias-

variance trade-off. Bias can be defined as the difference between the true value of a predicted outcome and the average predicted value by a model. High bias can arise due to the model giving erroneously higher importance to certain features in the dataset which in turn would result in a poorly trained model which performs badly both on the training dataset and testing dataset.

High bias is also commonly referred to as underfitting of a model to the data. Variance on the other hand refers to the ability of a model to fit to the spread of the data it is seeing during training, if this goes on uncontrolled then the model is said to be overfitting to the data and is not able to generalize well on the data it has not seen during training phase. So consequently the desired state of a machine learning model would be to have low bias and low variance. This motivates the needs for a trade-off between bias and variance where we want to give enough freedom to the model to explore underlying complexity of the data (in order to reduce bias) while also checking that too much freedom could lead to high variance.

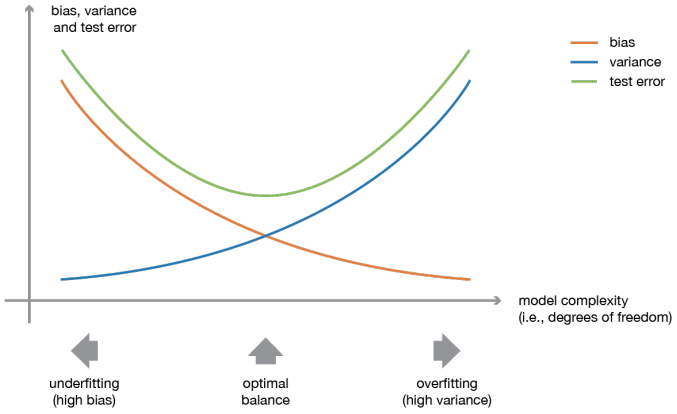


Figure 3.1: Bias-variance trade-off [16]

In ensembling, the bias-variance trade-off is catered to by combining multiple so called weak learners (low performance models) who are either having low bias and high variance or high bias but low variance. Based on how these weak learners are combined several different methods are used which are further described below:

1. Bootstrap Aggregating (Bagging) [1]: It combines the approaches of bootstrapping and aggregating. Bootstrapping is the method of generating new datasets equal to the size of the original dataset but with replacements. Any given bootstrapped dataset has at most only 63.2 per cent [9, Chapter 8] of unique samples. In bagging the overall focus is to have low bias, high variance weaker models which are balanced out by being combined together to make one stronger model.

   The bagging algorithm is given below [6]

   - Generate bootstrapped datasets.

- Train a model on each of the bootstrapped dataset.
- Average the models getting the bagging model

2. Boosting: Boosting based ensembling approaches lay emphasis on sequentially improving weak learners having low variance and high bias by learning from the mistakes of their predecessors. As we continue the process the high bias is eventually balanced out by improving on the mistakes of the previous learners.

3.2.1 Decision Trees

In order to create an ensemble of models there are no particular restrictions on the type of model that can be considered as a base model (weak learner), it can be a neural network or a decision tree or anything else for that matter. But in the scope of this thesis we are going to use decision trees as base models for implementing both bagging and boosting approaches. Thereby, in this section we will introduce the concept of a decision tree and further sections would explain how they can be used as base models to create ensemble models.

A decision tree also referred to as CART (Classification and Regression Trees) [3] can be understood simply as a structured hierarchical approach to reach a certain output given a particular set of inputs/attributes. A simple decision tree is shown in Figure 3.2, here we can observe a decision tree trained on the titanic datatset and we are predicting whether a passenger survived or not. We start process by checking the first node and depending upon the binary outcomes we traverse the tree until a terminal node is reached and that is our final prediction.

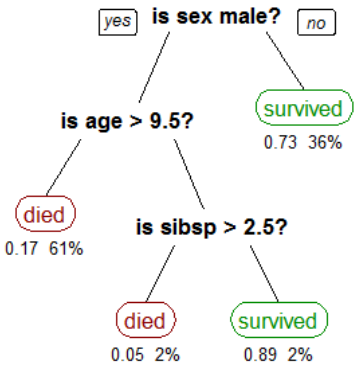


Figure 3.2: An example of a decision tree [14]

Decision tree learning is the process of selecting features and on those features we further select splitting points. The selection process is greedy and the aim in case of regression is to minimize the loss function (sum of squared error).

$$lossfunction = \sum_{i=1}^{N}(y_i - y_i')^2$$

The splitting of nodes is a recursive binary process which can go on even uncontrolled and would typically result in overfitting. One of the most commonly used stopping criterion is to assign a minimum number of training samples that can be used for each leaf nodes.

A single decision tree can often result in overfitting the training data. In order to stop this besides from using the stopping criterion we can also apply pruning on the trained decision tree. In the pruning process we evaluate the overall effect of leaf node on the loss function. This is done by removing leaf nodes one by one so that we can see which ones are affecting the loss significantly. This process of removing leaf nodes is continued until we are able to minimize the loss function and we stop once no significant improvements can be achieved.

3.2.2 Random Forests

When applying the bagging algorithm discussed earlier, if decision trees are being used as the weak learners then it gives us the random forests.

Formally it is defined by Brieman [2] as, *" Random forests for regression are formed by growing trees depending on a random vector such that the tree predictor $h(x, \Theta)$ takes on numerical values as opposed to class labels."*

It can be simply understood as a collection of decision trees whose output is averaged (for regression) and majority vote (for classification). A decision tree is a an individual decision stump in a random forest. Random forest algorithm is given below.

Algorithm 1: Random forest algorithm [6]

Input : N = Total number of trees, T is the original dataset, T_i is the bootstrapped dataset, Node size n_{min} is the number of dataset samples associated with the node

for *i = 1 to N* **do**

1. Draw a bootstrap dataset T_i from T, $|T_i|$ = m;
2. Grow a tree h_m^i using T_i by recursively repeating the following, until the minimum node size n_{min} is reached:
   a. Select k attributes at random from the p attributes
   b. Sick the best attribute and split-point among the k
   c. Split the node into two daughter nodes

end

Output : Ensemble of trees b(x) averaging h_i (regression) or selecting a majority vote (classification)

The decision to choose this algorithm was based on the following reasons:

1. It is well-known to provide high performance and also suits well to datasets of high dimensionality such as is the case with our dataset.

2. It controls overfitting on the training data which is a big disadvantage of using a single decision tree. In random forest, each decision tree is not given the choice of full features but rather a smaller subset at random and this in turn leads to difference in splitting attributes across the trees and overall we end up reducing the overfitting on the training dataset.

3. Random forests provide a computationally less expensive and faster way of error assessment by the use of out-of-Bag (OOB) error as compared to cross-validation (CV). The OOB is calcuated by[6]:

   - Produce boostrapped datasets from the original dataset.
   - For each observation in the original dataset choose only those trees from the random forest which were not trained on that particular sample.
   - To evaluate the error compute the average of OOB trees.

4. Random forest also provide two very useful metrics explained below which allow easy evaluation of feature importance in a dataset:

   a. Mean Decrease in Accuracy (MDA) or Permutation importance : It is the overall decrease in accuracy of the random forest due to removing a feature in the dataset. It can be calculated by using OOB error for the random forest. And then repeatedly permuting features and recording overall decrease in the accuracy.

   b. Mean Decrease in Impurity (MDI) or Gini Importance: It defined as *"the total decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)) averaged over all trees of the ensemble"*. [3]

3.2.3 Gradient boosting

Gradient boosting belong to a class of machine learning methods achieved by following boosting ensembling approach described previously in Section 2. The very basic idea here is to sequentially build weak classifiers which learn from the mistakes of the previous classifiers by boosting their gradient and reducing it for the incorrect classifiers. A weak classifier can be understood as something slightly better than a random guess. Hence, newer classifier learns from the mistakes of the previous classifier and this goes on until we achieve the classifier we need. The algorithm for the method from [7], [10] is given in Algorithm 2.

Algorithm 2: Gradient boosting algorithm [7], [10]

Input : training set $\{(x_i, y_i)\}_{i=1}^{n}$,$\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$,$L(y, F(x))$, number of iterations M.

Initialization *Initialize model with a constant value:*

$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$ **for** *m = 1 to M* **do**

1. Compute so-called pseudo-residuals:
   $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1, \ldots, n.$;
2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.;
3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)$$

4. Update the model: $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$;

end

Output : $F_M(x)$

In this thesis the focus has been on using XGBoost which stands for extreme gradient boosting and is described as *"a scalable machine learning system for tree boosting."*[5] It is a fairly new gradient boosting library which as per its name provides state of the art performance for boosting based ensembling methods and has proven to give good results especially when used with structured data. The library is a combination of superb advancements on both computational as well as some algorithmic aspects of its predecessors.

The library provides parallelization and effective use of memory resources during the training of models. A different pruning criterion of individual decision tree splitting has been introduced which is based upon limiting maximum depth of the trees instead of starting out with the traditional greedy approach. On the algorithmic side there have improvements in regularization of more complex models (both L1 & L2 regularization are now included) thereby, controlling overfitting. It has in-built implementation to allow cross-validation during the learning process and also enhanced sparsity awareness due to better handling of missing values through estimation based on loss in training. A weighted quantile sketch which helps in finding optimal splitting points amongst weighted datasets has also been added.

The decision to choose this algorithm was based on the following reasons:

1. The speed of learning is much faster compared to other boosting methods and is also further helped by the fact that it allows parallelisation. In a task like ours training on such big datasets would take a considerable

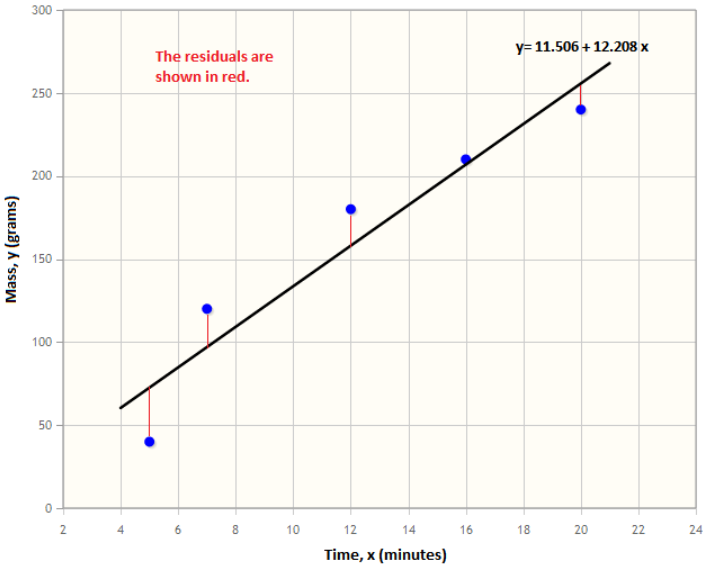


Figure 3.3: Residuals [18]

amount of time if classical implementations of gradient boosting trees were used as the model is built sequentially.

2. Based on its current reputation it is known to outperform other machine learning algorithms and thus, it would be interesting to analyze whether it would be the same case here.

3. Improved regularization can be very useful for controlling overfitting in our use-case.

3.3 Evaluation Metrics

In order to evaluate the performance of our models and to see their fit to the data we need to use certain metrics. Three widely used metrics used for regression analysis have been used here for evaluation [11].

Before we get to the metrics it is important to understand the concept of residuals which is used in the calculation of all the following metrics. A residual is defined as the difference between the predicted value and observed value of a quantity. It can also be understood as shown In Figure 3.3 as the distance between a data point and the line of fit of a model.

$$Residual = y_i - y_i'$$

All three metrics used for evaluation of the models are based on residuals and they follow the following convention where N is the total number of samples, y_i is the observed (true) value of a sample and y_i' is the predicted value of sample by the model, K is the total number of predictors.

- Root mean square error (RMSE)

  Mean squared error (MSE) refers to the square of the mean of the residuals.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2$$

  Root Mean Square Error refers to the square root of the mean of the residuals. It can simply be understood as the square root of the mean squared error (MSE).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2}$$

  The advantage of using RMSE over MSE is the ease in interpretibility since, it is represented in the same units as of the target variable. The value of RMSE can vary from 0 to ∞. Values close to zero indicate good fit of model and vice-versa.

- Mean absolute error (MAE)

  Mean Absolute Error refers to the absolute mean of difference of the residuals.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - y_i'|$$

  Characteristic feature of MAE over RMSE is that it normalizes the residuals, larger residuals would have bigger effect on the overall which MAE neutralizes due to taking absolute value of the difference. MAE is always non-negative and the smaller is the value the better is the fit of the model.

- R-squared and adjusted R-squared

  R-Squared metric is used for determining how well a particular regression model fits the data. It can also be understood as how well the variation in the target variable is explained by a model

$$R^2 = 1 - \frac{ExplainedVariation}{TotalVariation}$$

  The value of R-Squared metric varies from 0 to 1. The closer the value is to 1 the better fit is the model to the data.

  Since, the R^2-metric does not account for the number of predictors in the model for therefore, we use the Adjusted R-squared metric. Which enables comparison of models with different predictors.

$$Adjusted R-Squared = 1 - \frac{(1-R^2)(N-1)}{N-K-1}$$

The value of Adjusted R-Squared metric varies from 0 to 1. The closer the value is to 1 the better fit is the model to the data.

Chapter 4

Data pipeline

In this section we describe the semi-automated data pipeline created for data collection, processing and further preparation for analysis and training of machine learning models. A high level diagram is shown below and individual components of the pipeline are explained in following subsections.

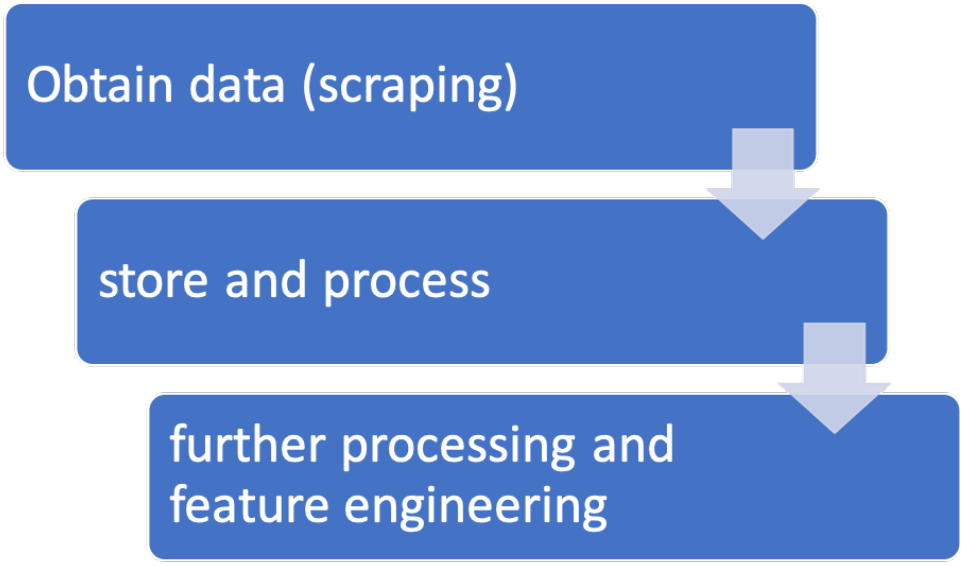


Figure 4.1: Data Pipeline

4.1 Data collection methodology

The data used in the thesis was collected from the websites of the two bus service providers namely Student Agency and FlixBus. It was scraped for buses departing during the period from 11th of October 2018 till 2nd of January 2019 and 20th of March 2018 till 5 December 2019 for FlixBus and Student Agency respectivly. It took us a long sequence of steps from starting with inspecting the websites to figuring out which fields provide potentially useful data so that specific scrapers could then be developed. Separate scrapers using python programming language were developed for

each of the service provided as their websites differ from each other which can also be observed in the Figure 4.2. These scrapers were then run at regular periods to obtain raw data which is stored in varying formats and processed further.

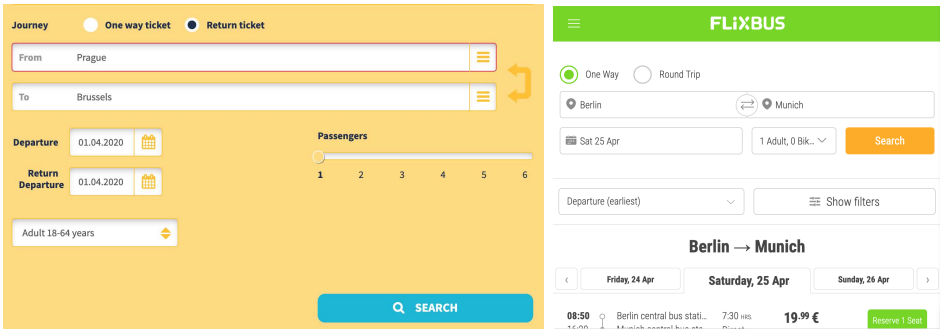


Figure 4.2: Websites of the service providers

The scripts scraped the data from the websites of the service providers, storing each sample separately in individual files. The individual files each represent a single scraped record which are further combined together using scripts in Python.

4.2 Data Cleaning

Once we have the individual records combined as a single file, then the next step is to clean them. The data at hand contains many issues such as unparsable data formats, misaligned data fields, missing data in certain columns along with many other unwanted things that have crept in during the scraping and further combination of individual records. Before the data could be analyzed, it is necessary to deal with above mentioned inconsistencies. For performing the cleaning operations scripts were prepared in Python. Cleaning results in loss of certain amount of data (approximately 2 - 5 per cent) as we discarded data having missing values.

4.3 Feature engineering and transformation

Now that we have cleaned datasets available, the next step in the data pipeline involves generation of certain new features from existing ones (which are then ultimately removed) while some are transformed from one format to another. The motivation behind this is described below:

- To reduce the number of features or dimensionality of the datasets: as the number of features increase, the model could become more and more complex and it also increases the chances of overfitting to the training data.

- Quality of model: presence of redundant variables results in misleading predictions and a low quality model. More the number of useful features the better it is for the model to learn the underlying relations.
- Less computations: a reduction in number of features would also result in faster computations and hence, would speed up the training of models.

The following new features were genereated and added to the datasets based on the pre-existing features:

1. Day of travel - A categorical text based feature detailing the day of the week the connection is operating. The feature which is present in both the datasets was generated based on the date of departure feature which is in form of time and date stamp. Getting days of travel is a very useful feature as it could help us understand how the frequency of buses is altered by the service providers.
2. Distance - this feature was explicitly added using the route column in the datasets. Based on the route we used Google maps API to find distance travelled between two cities.
3. Hours before departure - this feature was obtained by using two pre-existing features in the datasets namely date of departure and scraping date. It was obtained as the difference between scraping time and time of departure. By generating this new feature we could remove scraping data and departure data and it resulted in reducing the number of features and the biggest advantage is that having number of hours is a numerical and hence, easily machine readable while scraping and data are date and time stamps which are treated as categorical variables by the ML algorithms and it would complicate the model further and we would end up one hot encoding them which would add nearly 20 columns more to our training datasets.

Several variables were transformed from their initial formats to easily machine readable formats in order to help the ML algorithms understand them better and also for easier analysis.

1. Columns departure time and arrival time of buses which are originally in 24 hours time format were transformed into decimal format by using a simple created function in R. If this conversion was not done the model would treat this data as categorical variables and one hot encoding would result in adding up to 50 variables into the training datasets.
2. Column transfer in the Flixbus dataset which was initially categorical with two levels - transfer and without transfer was recoded into binary format with 0 indicating without transfer and 1 indicating a connection with transfer.

Chapter 5

Dataset Description

In this section the datasets obtained after the passing through the data pipeline are discussed. These are the final datasets on which we conduct further analysis as well as our machine learning models will be trained. But before we get to statistics below we have defined certain terms used in the thesis from this section onwards:

- Sample: refers to a single scraping record of a bus. There can be multiple samples for the same bus for the same route.
- Connection: refers to only a particular bus departing at a particular date and time. A connection is up made of multiple samples.
- Route: refers to the source and destination cities of a bus. A route can have multiple connections.

Dataset statistics	Student Agency	FlixBus
Total samples	3803458	1252067
Total connections	10	16
Missing cells (%)	0.0 %	0.0 %
Duplicate rows (%)	0.0 %	0.0 %
Number of variables	10	10
Numerical variables	7	6
Categorical variables	3	2
Boolean variables	0	1

Table 5.1: Overview of the datasets

The final cleaned datasets are having a total 3.8 million samples for Student Agency and while for FlixBus we have 1.3 million samples. The reason why the samples are much less for FlixBus dataset is because it was scraped much less as compared to Student Agency. Furthermore the data that was scraped had a large of number of duplicates (nearly 50 per cent) present in it which was discovered only after running an early exploratory analysis and as a result these samples were subsequently dropped. A more detailed overview can be seen in Table 5.1 .

Feature Name	Description	Type
col_depart	departure time for a bus from the source	Numerical
col_arrival	arrival time for a bus to the destination	Numerical
col_space	number of seats free for booking in a bus	Numerical
col_price	price of single ticket in the bus	Numerical (in CZK)
distance	distance travelled by the bus in travelling from source to destination	Numerical (in kilometers)
travel_time	time taken by the bus to travel from the source to destination	Numerical (in hours)
transer	whether the journey involves a transfer	Boolean
hours_before_departure	time period between scraping time of data and the departure date and time of the bus	Numerical (in hours)
day_of_travel	day of departure of the bus	Categorical
services	For student agency we have different types of services being offered. There are total 8 different types of service categories which have been numbered accordingly from 1 to 8	Categorical

Table 5.2: Features in the datasets

The final processed datasets are having a common format where they share nearly the same of features. Majority of them are general features which can be expected in any transport related dataset such as departure/arrival times, number of passengers, distance to travel (covered by the departure point and arrival point), price of a ticket, day of the week the travel is taking place and available free space in the bus.

But there is one unique feature in the Student Agency dataset which is termed as service type which sets it apart from only from the Flixbus dataset. Service type is a categorical variable which is referring to eight different categories of services offered by Student Agency on board its buses.

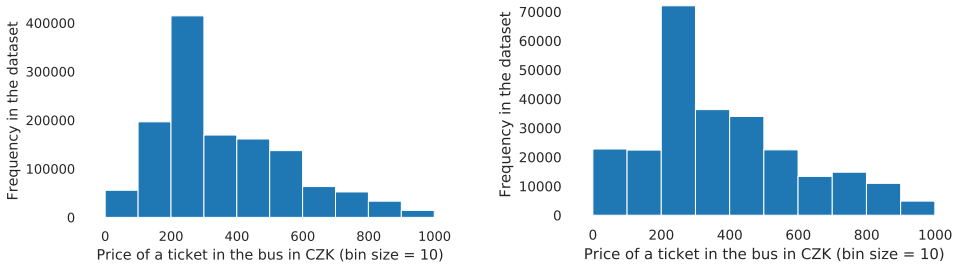
Chapter 6

Analysis of Datasets

In this section the features described in the previous section are analyzed in order to present a view of the current situation, derive insights and formulate a better understanding of the data which is needed before we start to train our machine learning models.

6.1 Feature distributions

We start off by analysing how the various features of interest are distributed across different datasets. This is important from the point of view of further analysis as it could possible help us detect some class imbalances happening in the data. If we conduct analysis on unbalanced datasets then that could give us skewed results and furthermore, it could also lead to serious errors when training machine learning models.

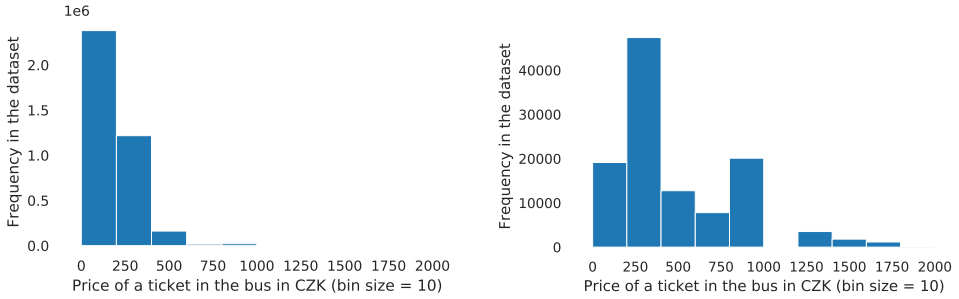


(a) : Original dataset **(b) :** Reduced dataset

Figure 6.1: Distribution of price feature in the FlixBus dataset

The distribution of prices in the FlixBus dataset as observed in 6.1a is quite evenly spread overall except for a large peak in the range 200 - 300 CZK. But as we analyse the distribution for the free capacity we discover a huge imbalance as shown in Figure 6.3a where it reveals the maximum observations are in the range 35-40 in the data and other bins of the histogram are almost invisible.

This imbalance is likely due to the dataset used for FlixBus. Since we were collecting data for periods up to 3 months before the departure of buses and it is likely the buses are not fully occupied that early so as a result we

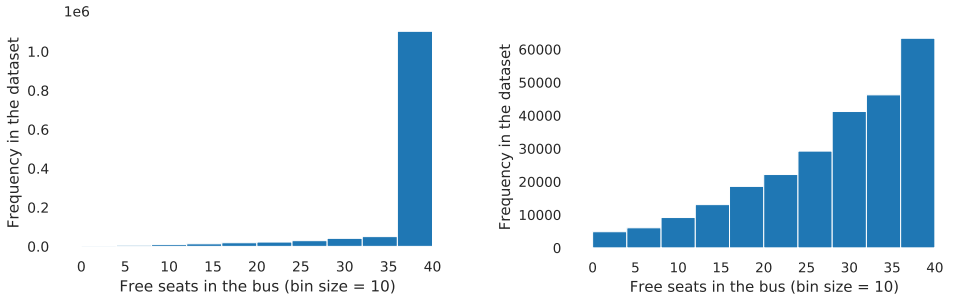


(a) : Original dataset **(b) :** Reduced dataset

Figure 6.2: Distribution of price feature in the Student Agency dataset

have many observations where the free capacity is very high. Besides this another major reason of having much more samples in this range is during the scraping it was not possible to record a free capacity of more than 40. So as a result in records where the free capacity was even more than 40, it was still recorded as 40.

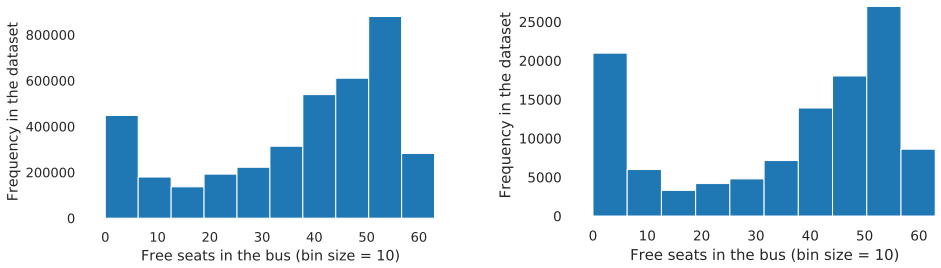
It is important to balance the distributions here otherwise our models for example will just learn to predict the free capacity in the ranges of 35-40 and other classes will be totally ignored. For balancing out the dataset, we randomly undersampled the rows of data where capacity is between 30-40. The resulting distribution is represented in Figure 6.3b where we can now observe prices in ranges outside 30-40. Interestingly, the overall distribution of prices though still remains more or less unchanged as can be in seen in Figure 6.1b.



(a) : Original dataset **(b) :** Reduced dataset

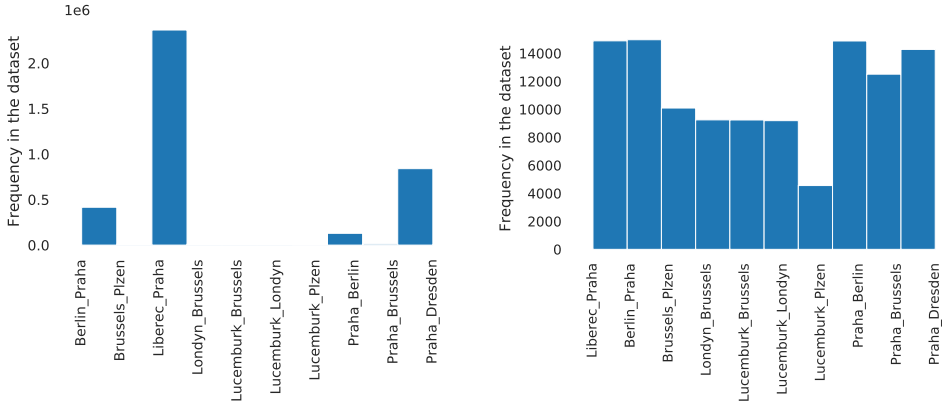
Figure 6.3: Distribution of free space feature in the Flixbus dataset

For the Student Agency dataset, as soon as we plot the price ranges we discover a huge imbalance shown in Figure 6.2a where majority of the observations in the dataset are in price range 0-500 CZK. The cause of this imbalance in the Student Agency dataset stems from the disproportionate presence of connections which is shown in figure 6.5a, where we can see the number of samples in the dataset for each connection. On the first look, one can observe the largest group of samples is from Liberec to Prague, Prague to Berlin (both ways) and Prague to Dresden connections while other



(a) : Original dataset **(b) :** Reduced dataset

Figure 6.4: Distribution of free space feature in the Student Agency dataset



(a) : Original dataset **(b) :** Reduced dataset

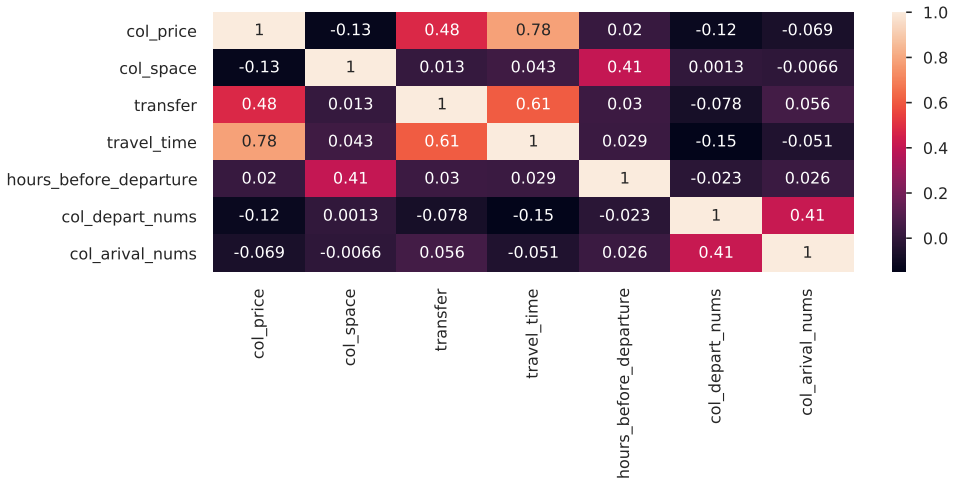
Figure 6.5: Distribution of connection feature in the Student Agency dataset

connections are almost invisble in the figure. Liberec to Prague connection is usually having the prices in the set range of 79, 89 and 99 CZK whereas, the connections to Germany from Prague are usually priced from 250 - 500 CZK range. Hence, this is the reason why the biggest bins in our price histograms are in that range. This disproportionate representation of connections is possibly due to a combinations of two factors - firstly, the number of connections on the routes mentioned before is very high and as a result more samples were collected during scraping of data. Secondly, it could be due to the fact certain connections and routes were scraped more during the data collection process.

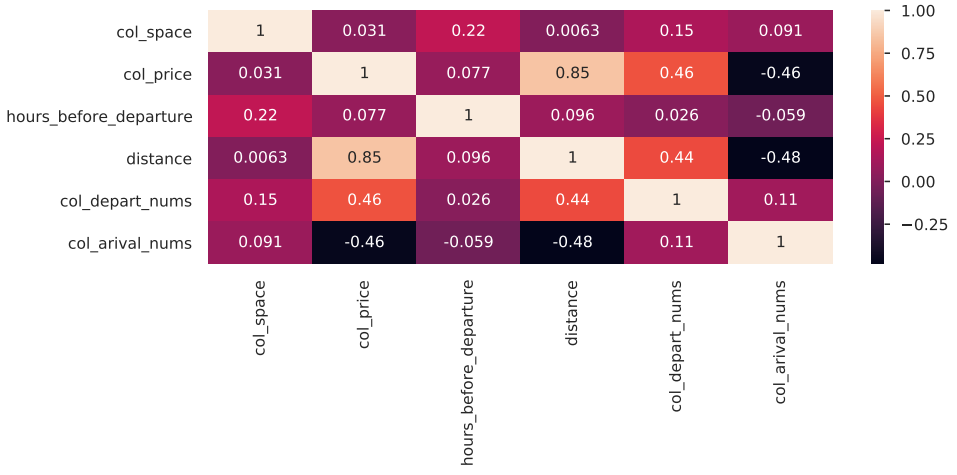
For balancing the connections we undersampled high frequency connections reported earlier and new distributions can be observed in Figures 6.5b and 6.2b for price and connection frequency respectively. The price is now more evenly distributed across the different ranges. This also helps to ensure that models that will be trained on the reduced amount of data learn the underlying relationships of target variables across different connections. If certain connections overpower the dataset then it is highly likely the models will overfit to those specific connections. The balancing of connections does not seem to effect the distribution of free capacity feature in the dataset and it

remains more or less unchanged after the data is reduced. On a comparative note to the FlixBus dataset, the original Student Agency dataset shown in Figure 6.4a does not show a huge disparity in distribution of free capacity feature. A possible reason why this may be happening is because the samples for the this dataset were collected over a much longer period (approximately one year) as compared to the FlixBux dataset (approximately three months). Longer period of data collection meant more representation of various ranges in the dataset.

6.2 Assessing relationships among the features



(a) FlixBus dataset



(b) Student Agency dataset

Figure 6.6: Correlation matrices for the datasets

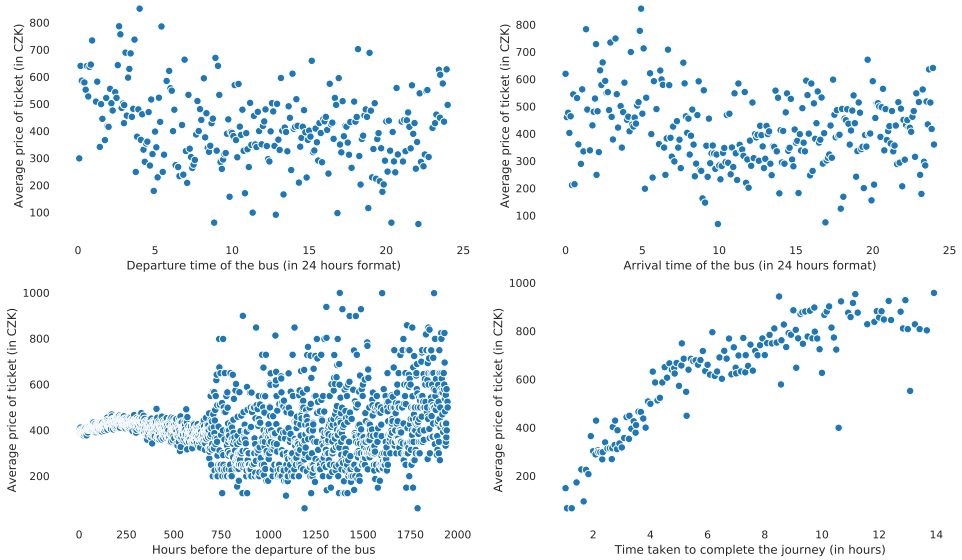


Figure 6.7: Average price of versus all other numerical variables in the FlixBus dataset

Having assessed the features distributions and corrected imbalances in the datasets we can proceed with further analysis of how the features are interacting with each other. Observing correlation matrices for the datasets in Figure 6.6, we can see almost all features except distance and departure time are weakly correlated to price for Student Agency dataset. While for FlixBus dataset similar scenario is depicted by two features namely travel time and transfer. A weak correlation could possibly indicate existence of either a non-linear relationship or certain features are just redundant and have very little effect on price overall. This is something we will discover later through our predictive analysis.

A similar scenario can be observed for free capacity as well, where both datasets show very similar correlation scores and the only feature which seems to show a strong relationship with the free capacity is hours before departure of the bus. This could again point towards the existence of non-linear relationships among the features or could simply be a case of having surplus features in the datasets.

Now that we have a slight hint of overall feature interactions through the correlation matrices, for a more deeper understanding of feature relationships we will use pairwise plots of average price and free capacity with respect to all other numerical features in the datasets.

In Figure 6.7 we can observe how the average price of a ticket is being affected by other features in the FlixBus dataset. As pointed out by the correlation matrix earlier, the relationship of price seems to be strongly linear with respect to travel time. As for the remaining three numerical features the relationship of average price with respect to hours before departure shows a slightly linear trend when the hours before departure is less than 250 hours but beyond that point the relationship is much harder to interpret directly

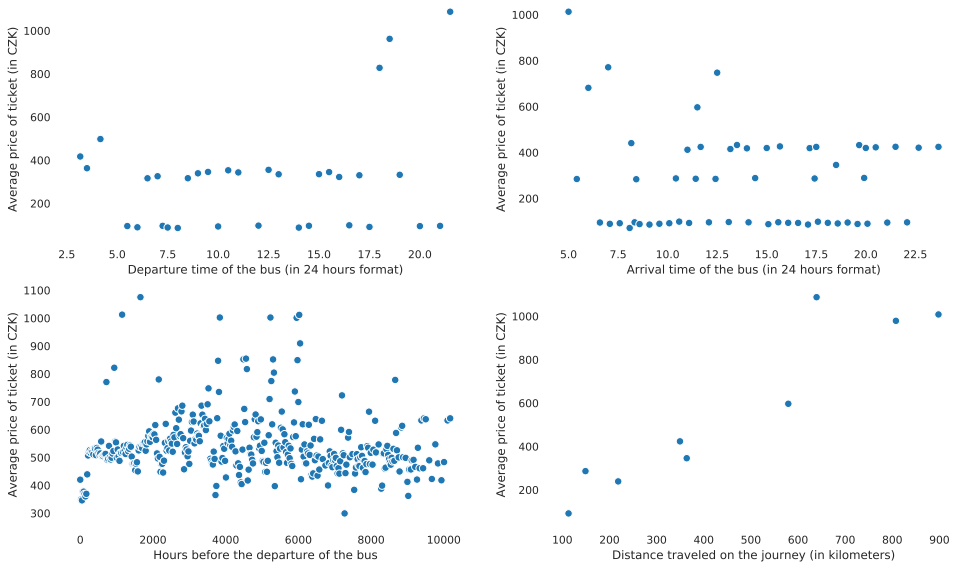


Figure 6.8: Average price versus all other numerical features in FlixBus dataset

from the plot and seems to be strongly non-linear. For the departure and arrival times of the bus, the distribution of points seems to be quite evenly spread and we can observe average price is slightly higher for bus departing during midnight and early morning hours.

For the Student Agency dataset in Figure 6.8 except for distance traveled on the journey all other features seem to be non-linearly related to average price of the ticket. Although in comparison to the FlixBus dataset, we can observe the fluctuation in average price is much less with respect to the departure and arrival times of the bus.

The relationship of average free capacity with other features in the FlixBus dataset can be observed as being quite non-linear. We can see in Figure 6.9, the relationship with departure and arrival times is similar to average price. But the interactions with hours before departure seem to be hyperbolic and we can observe the average free capacity is dropping quite rapidly as the hours before departure are decreasing. The situation is slightly different for the average free capacity in Student Agency dataset as shown in Figure 6.10. There is a slight hint of some linear relationship with departure times but arrival times show a mostly a non-linear trend. Also the relationship with hours before departure is existing over a much larger range and the hints of a similar hyperbolic trend as seen in the FlixBus dataset can be noticed here as well.

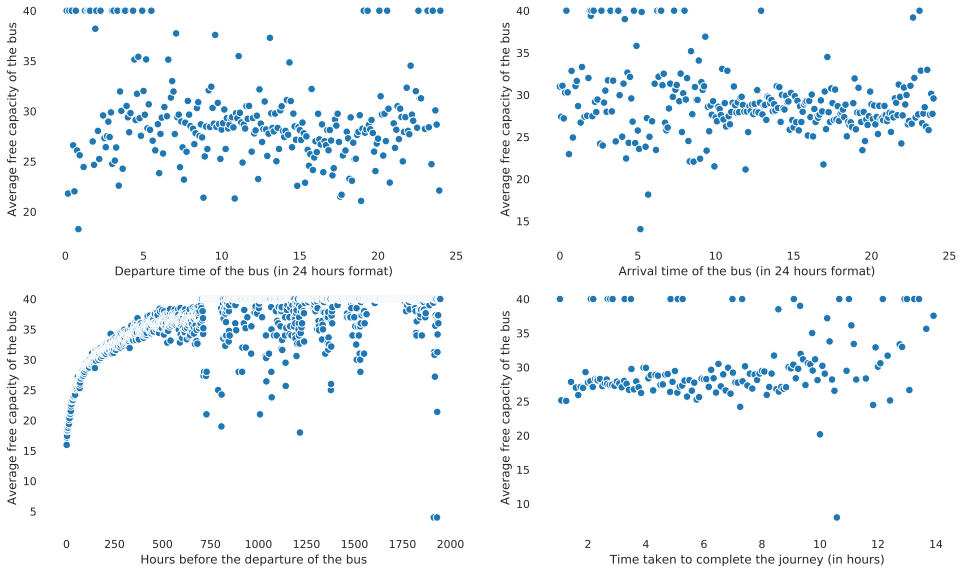


Figure 6.9: Average free capacity versus all other numerical features in FlixBus dataset

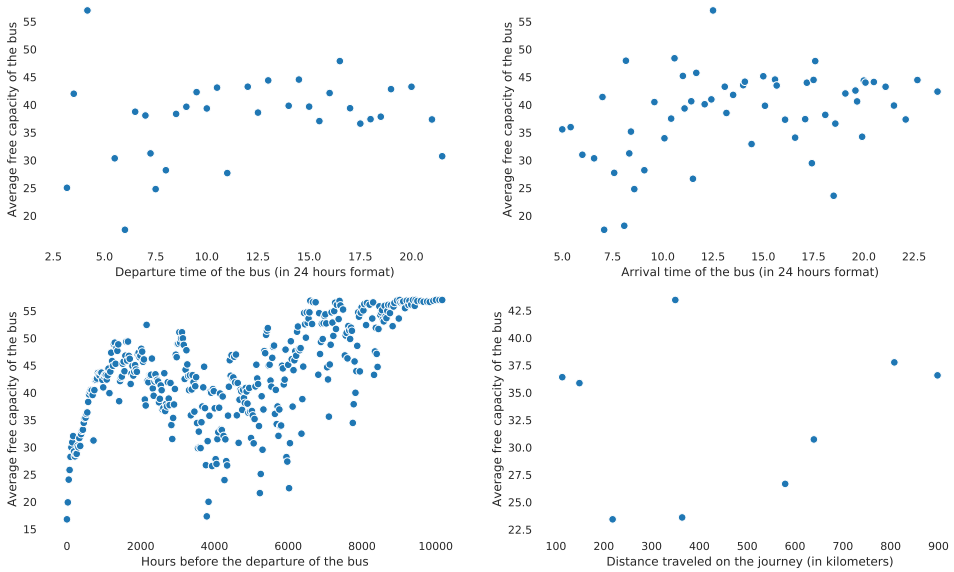


Figure 6.10: Average free capacity versus all other numerical features in Student Agency dataset

Chapter 7

Training of models

This chapter covers the application of the machine learning techniques described in Section 3.2 to the datasets. We will be training separate models for predicting both price and the free capacity in a bus. The entire process of training, tuning and testing of the machine learning models is described in further subsections along with the respective evaluation metrics discussed in Section 3.3 to evaluate their performance.

Before moving further we would present certain terminology along with it's meaning which is used in this section and also further beyond:

- Training set: Refers to a partition of the original dataset which is used to train a model.
- Testing set: Refers to a partition of the orignal dataset which is unseen by the model during its training phase. It is used to evaluate the performance by assessing it's ability to generalize over data it has not seen during training.
- Model hyperparameters: Refers to those parameters of the model which cannot be learned during its training and are required to be set at the start of the training process.
- Validation set: Refers to a partition of the orginial dataset which is used to tune the hyperparameters of a model. The validation set may or may not be data the model has seen during training.
- Cross-validation: It is the technique of partitioning the original dataset into different partitions and the model is trained on a partition of data while the remaining is used as a testing set to evaluate the performance. This approach of partitioning the datasets is more specifically called k-fold cross validation where k refers to the number of partitions created from a dataset. The dataset is divided into k-folds then the model is trained on k-1 folds always leaving one partition out to be used as a testing set, the whole process is repeated total k times. There are also other cross-validation techniques but this is only one being used for our experiments in the thesis.

- Grid search: Refers to the process of tuning of hyperparameters of a model where we select different ranges of values to be tried out and then combinations are made of all the values and search is performed to find the best hyperparameters. Grid search also has different types but in the scope of our work we have used a constrained version of a full grid search where we take only a pair of hyperparameters and tune them at a time. This contrained version was used in order to optimize the use of computational resources.

- One-hot encoding: Refers to the technique of transforming categorical variables in a dataset to a binary representation. It allows easier understanding of categorical variables by machine learning models.

7.1 Baseline model: Linear regression

In order to understand how well our trained models are doing, it is important to have a baseline to compare them against. The baseline model can be any weak model which is slightly better than a random guess. Such a model can be the average of the prediction variable or an ordinary least square (OLS) regression also known as linear regression which is what we will use as the baseline model. Evaluation metrics have been presented in the Tables 7.1 and 7.2 for the baseline linear regression models for price and free capacity predicting models respectively for both Student Agency and FlixBus.

	Price Model		Free Capacity Model	
Metrics	Training Set	Testing set	Training Set	Testing set
RMSE	169.68	169.91	17.89	17.92
MAE	94.83	94.97	15.14	15.11
Adjusted R-Squared	0.60	0.60	0.03	0.03

Table 7.1: Linear regression results for Student Agency Dataset

	Price Model		Free Capacity Model	
Metrics	Training Set	Testing set	Training Set	Testing set
RMSE	96.41	96.45	8.07	8.05
MAE	67.10	67.21	6.29	6.28
Adjusted R-Squared	0.64	0.64	0.078	0.077

Table 7.2: Linear regression results for FlixBus Dataset

From the tables above we can already observe the linear regression model is not performing too well. If we take a closer look at the evaluation metrics

presented, RMSE of 169.68 for example can be understood as on an average the output of the model is off from the true value by 169.68 CZK. It is a significant error in the context of our scales of price (0-2000 CZK) but that is expected since, this is just a baseline model.

The key feature in the tables here though is the adjusted r-squared metric which is indicating a moderately high value of fit of the price linear regression model to both Student Agency and FlixBus data. On the other hand, if we observe the adjusted r-squared metric presented in Tables 7.1 and 7.2 above we will notice the value is very low for free capacity linear regression models for both the datasets. This can possibly indicate two things, one the there is a requirement of a non-linear model here. The second thing it could indicate also is if we refer back to definition of adjusted r-squared we are also taking into account the number of features and a low adjusted r-squared indicates there are too many redundant independent variables in the model.

7.2 Random Forest Models

As we have the baseline metrics set, we can start training our machine learning models. We will start by training the random forest models for predicting both the price and free capacity in a bus. For implementing the algorithms and developing our models we have used the sklearn [15] package in Python.

Preparation of data

One of the key features of random forest algorithm also mentioned previously is that it does not require extensive feature pre-processing to be applied on the data before the model can be trained. Only transformations required to be done were one-hot encoding of categorical variables in the datasets. After that we split the entire dataset into training and testing set in the ratio of 75:25, for performing this split we use random sampling provided by sklearn in Python.

Training of random forest models and hyperparameter tuning

Random forest much like many other machine learning algorithms is dependent on hyperparameter tuning to give the best results. Building a random forest model is usually an iterative process where we start out by building a baseline random forest model which is then further tuned to give the best performance. Although there are many hyperparamaters in the python implementation of random forests in the sklearn package which can be tuned but below we have mentioned some of the most important ones along with their short definitions, which were used and tuned in our work:

- *max_depth*: refers to the maximum depth up to which a single tree is allowed to grow.

- *n_estimators*: refers to the number of trees to be used in the random forest.
- *min_samples_split*: refers to the minimum number of samples which must be present for a node to be split.
- *max_samples*: refers to fraction of the bootstrapped samples to be used for training each tree in the random forest.
- *max_features*: refers to the number of features to be considered while searching for the best split criterion for nodes in each tree in the random forest.

Hyperparameter	Tuning Grid	Best value (price model)	Best value (free capacity model)
max_depth	[3,5,7,9]	9	9
n_estimators	[50,200, 500, 1000]	500	1000
min_samples_split	[2,4,6,8,10]	2	4
max_samples	[0.50,0.65,0.80]	0.80	0.80
max_features	['auto','sqrt','log2']	auto	auto
Baseline RMSE		102.58	7.25
Best tuned RMSE		77.13	5.72

Table 7.3: Random forest hyperparamter grid tuning results for FlixBus dataset

Hyperparameter	Tuning Grid	Best value (price model)	Best value (free capacity model)
max_depth	[3,5,7,9]	9	9
n_estimators	[50,200, 500, 1000]	50	500
min_samples_split	[2,4,6,8,10]	2	10
max_samples	[0.50,0.65,0.80]	0.50	0.50
max_features	['auto','sqrt','log2']	auto	auto
Baseline RMSE		101.31	15.50
Best tuned RMSE		73.65	13.21

Table 7.4: Random forest hyperparamter tuning grid results for Student Agency dataset

Following the iterative process mentioned earlier we begin by training base random forest models for both price and free capacity on both the datasets separately. These base models are trained using default settings provided by the sklearn package and there performance as expected is not quite good, the exact figures can be observed in Tabls 7.3 and 7.4 respectively. We then apply

constrained grid searching combined with 5-fold cross-validation to tune the hyperparameters of our base models and this further results in significant gains in metrics for all the base models. Through this tuning process we were able to lower the RMSE of the price models by approximately 20 CZK and for the free capacity models further by 2 seats.

Evaluation of final trained models

Once we have obtained the best tuned values for the hyperparameters, final models are trained and their performance is evaluated using the testing set. The metrics are presented in the Tables 7.5 and 7.6. A quick glance shows there is not much disparity between training metrics and testing metrics thereby, indicating the models are not overfitting.

	Price Model		Free Capacity Model	
Metrics	Training Set	Testing set	Training Set	Testing set
RMSE	75.81	77.13	5.65	5.72
MAE	7.47	7.53	4.32	4.37
Adjusted R-Squared	0.86	0.86	0.41	0.41

Table 7.5: Final random forest results for Flixbus Dataset

	Price Model		Free Capacity Model	
Metrics	Training Set	Testing set	Training Set	Testing set
RMSE	73.65	76.05	13.0	13.21
MAE	29.82	30.23	9.25	9.38
Adjusted R-Squared	0.95	0.95	0.32	0.30

Table 7.6: Final random forest results for Student Agency Dataset

7.3 XGBoost Model

Having finished training our random forest models, now we move on to the second class of ensemble techniques based around boosting described in section 3.2 and more specifically using the XGBoost library. For training and tuning of our XGBoost models we have used the XGBoost python package [5] along with it's sklearn API version. The reason we used a combination of these two packages is to use the easy-to-use and efficient grid searching capabilities for hyperparameter tuning provided by the sklearn package.

Preparation of data

The process of preparation of data for training for the XGBoost model is very similar and almost identical to the one mentioned previously for the random forest models. Except that XGBoost has a strict requirement to use only numerical features which required one-hot encoding of categorical variables in the datasets. This is further followed by transformation of the datasets in python from dataframes to an internal data structure called as DMatrix used by the XGBoost library.

Training of XGBoost models and hyperparameter tuning

Similarly to random forests, XGBoost is also heavily reliant on hyperparameter tuning to give the best results. Hyperparameters were tuned using the sklearn package's grid search combined with 5-fold cross validation. A short description of these hyperparameters as described by the documentation of the packages is given below:

- *max_depth*: refers to the maximum depth up to which a single tree is allowed to grow.
- *min_child_weight*: refers to the minimum sum of weight needed in a child node.
- *gamma*: refers to the minimum reduction in loss which must occur in order to partition a leaf node in the tree.
- *subsample*: refers to the subsample ratio of training data to be used.
- *colsample_bytree*: refers to the subsample ratio of training data to be used when growing each tree in the ensemble.
- *reg_alpha*: refers to the L1 regularization on weights during training. A very useful parameter to control overfitting.

We follow the same iterative process described earlier for random forests i.e. building base models using default settings in the sklearn package and further tuning the hyperparameters using grid search and 5-fold cross validation. The applied grid search values along with the best values can observed in Table 7.7 and 7.8 for the FlixBus and Student Agency datasets respectively. The gain from hyperparameter tuning is quite significant here especially in the case of price models where the RMSE was significantly lowered for the FlixBus dataset from the base XGBoost model value of 143.40 to best tuned value of 42.91. Similar results are obtained for the Student Agency dataset as well, where we lowered the RMSE by approximately 30 CZK from base model value of 86.86. This clearly shows the value of the hyperparameter tuning process which although is quite time consuming and computational resource intensive but reaps good results.

Hyperparameter	Tuning Grid	Best value (price model)	Best value (free capacity model)
max_depth	[3,5,7,9]	9	9
min_child_weight	[1,3,5]	3	1
gamma	[0.0, 0.1, 0.2, 0.3, 0.4]	0	0
subsample	[0.6, 0.7, 0.8, 0.9]	0.8	0.9
colsample_bytree	[0.6, 0.7, 0.8, 0.9]	0.9	0.9
reg_alpha	[1e-5, 1e-2, 0.1, 1, 100,1000]	1	0.1
Baseline RMSE		143.40	5.40
Best tuned RMSE		42.91	4.11

Table 7.7: XGBoost hyperparamter grid tuning results for FlixBus dataset

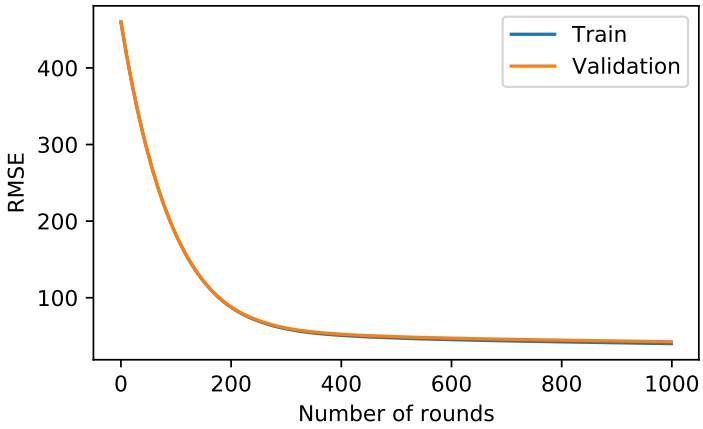
Hyperparameter	Tuning Grid	Best value (price model)	Best value (free capacity model)
max_depth	[3,5,7,9]	9	9
min_child_weight	[1,3,5]	1	1
gamma	[0.0, 0.1, 0.2, 0.3, 0.4]	0	0
subsample	[0.6, 0.7, 0.8, 0.9]	0.9	0.9
colsample_bytree	[0.6, 0.7, 0.8, 0.9]	0.9	0.6
reg_alpha	[1e-5, 1e-2, 0.1, 1, 100,1000]	100	100
Baseline RMSE		86.86	13.23
Best tuned RMSE		59.22	11.18

Table 7.8: XGBoost hyperparamter grid tuning results for Student Agency dataset

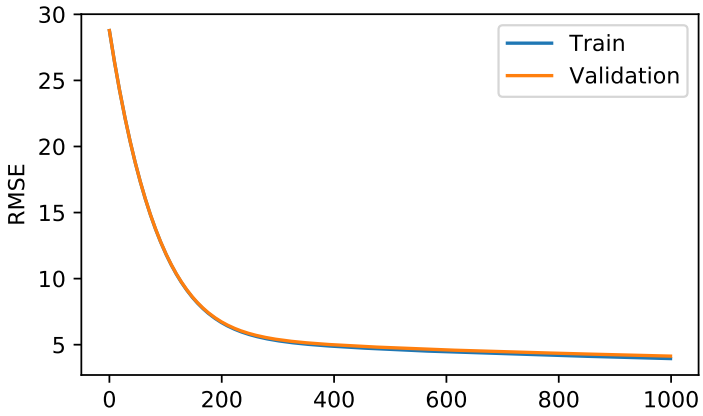
Evaluation of final trained models

Having determined the best value of hyperparameters now we can proceed with the training of final models. XGBoost library is used to perform the training process for one thousand rounds with 10-fold cross-validation to achieve more robust models combined together with early stopping and lowered learning rate. Training results can be be observed in Figure 7.1 for the FlixBus models and we can observe training and validation curves are following each other very closely. This demonstrates we are not overfitting and also we can understand why smaller values of regularization alpha (1 and 0.1) are given in Table 7.7.

While on the other hand, the plots given in Figure 7.2 are slightly different



(a) Price model



(b) Free capacity model

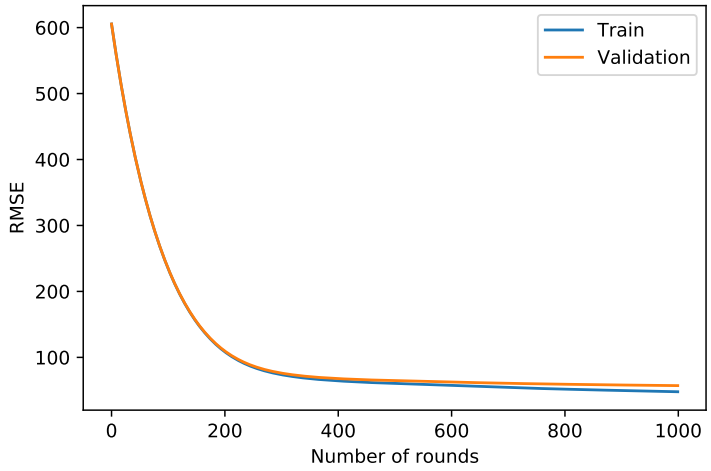
Figure 7.1: Final model training errors for FlixBus dataset

for the Student Agency models. Here the curves for training and validation errors are starting to diverge from each other after six hundred and two hundred rounds for the price and free capacity models respectively. This departure of curves from each other is usually an indication that the model is now beginning to overfit the data and hence, this explains why we can see much higher values (100 for both models) of regularization alpha in Table 7.8. The alpha or the L1 regularization helps in controlling overfitting by highly penalizing more complex models.

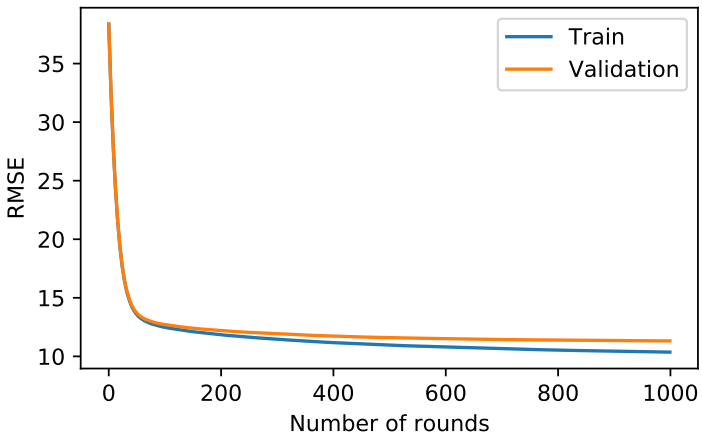
After completing the training process, the testing set is used to evaluate the models' ability to generalize. As we do not see much difference between the training and testing errors in tables 7.9 and 7.10 for both FlixBus and Student Agency models, it indicates the models have learned well.

Metrics	Price Model Training Set	Testing set	Free Capacity Model Training Set	Testing set
RMSE	40.13	42.91	3.96	4.11
MAE	28.88	30.47	2.97	3.09
Adjusted R-Squared	0.93	0.92	0.67	0.65

Table 7.9: Final XGBoost results for the Flixbus dataset



(a) Price model



(b) Free capacity model

Figure 7.2: Final model training errors for Student Agency dataset

Metrics	Price Model Training Set	Testing set	Free Capacity Model Training Set	Testing set
RMSE	49.84	59.22	10.36	11.18
MAE	21.71	24.19	6.89	7.44
Adjusted R-Squared	0.97	0.94	0.53	0.46

Table 7.10: Final XGBoost results for the Student Agency dataset

Chapter 8

Evaluation and Interpretation of Results

In this section we will evaluate the fit of the various models to the datasets in order to determine which models are performing well and which ones are not. Once we have determined our best models, we will further interpret their output and along with the relationship depicted by the output variables with their respective inputs. This would further enable us to understand the major features influencing the price and free capacity in the datasets.

8.1 Comparing the methods based on evaluation metrics

Model	Price Model RMSE	Adjusted R-Squared	Free Capacity Model RMSE	Adjusted R-Squared
Linear regression	169.91	0.60	8.05	0.07
Random forest	77.13	0.86	5.72	0.41
XGBoost	42.91	0.92	4.11	0.65

Table 8.1: Evaluation metrics for FlixBus models

For our price models we can observe significant improvements for both FlixBus and Student Agency in Tables 8.1 and 8.2 respectively. XGBoost has quite clearly outperformed both the baseline as well as the random forest models by having an RMSE of 42.91 and 59.22. Which can be interpreted as on average the output of the model is off from the true value by 42.91 CZK. A high value of adjusted r-squared also shows that the model is fitting to the data quite well. This is not very surprising as with the baseline model we already had a good value of fit 0.60 so having a non-linear model this was expected to increase.

For the free capacity models also XGBoost has once again outperformed its competitors by producing the lowest RMSE for 11.18 and 4.11 reported in

Tables 8.1 and 8.2 respectively. Also there is a quite significant improvement in the fit of the model to the data as indicated by values of 0.65 and 0.46 from base values 0.07 and 0.03 respectively. This further reinforces our observations in Section 6.2 that the relationship is strongly non-linear in case of free capacity in the datasets. It also indicates that are not enough features in the dataset to accurately predict the free capacity.

	Price Model		Free capacity Model	
Model	RMSE	Adjusted R-Squared	RMSE	Adjusted R-Squared
Linear regression	96.45	0.64	17.89	0.03
Random forest	76.05	0.95	13.21	0.30
XGBoost	59.22	0.94	11.18	0.46

Table 8.2: Evaluation metrics for Student Agency models

8.2 Interpretation of the trained Models

Interpreting the outputs of machine learning methods such as random forest and XGBoost is an active area of research in the machine learning community. Over the years some reliable approaches have become best practices in the industry which are applied in the sections to come to interpret the output of the models trained earlier.

Global interpretations

In this section we will evaluate our models on a global level i.e. for the entire datasets. The key technique for this for ensemble models is through evaluation of feature importances. Feature importance is a key feature which comes as an out-of box component of most of the conventional libraries based around ensembling techniques. It evaluates the importance of each independent variable in the dataset with regards to the dependent variable. The concept here is based on the concept of mean decrease in impurity (MDI) defined earlier in Section 3.2. Uses of feature importance:

- Feature selection: Using feature importance we can evaluate which are the most important in making better predictions and also the ones which are least important. This can further help us to remove these less important features thereby, making our model simpler.
- Provides a basis of cross comparison: feature importance is computed by both the random forest and XGBoost methods thereby, providing a

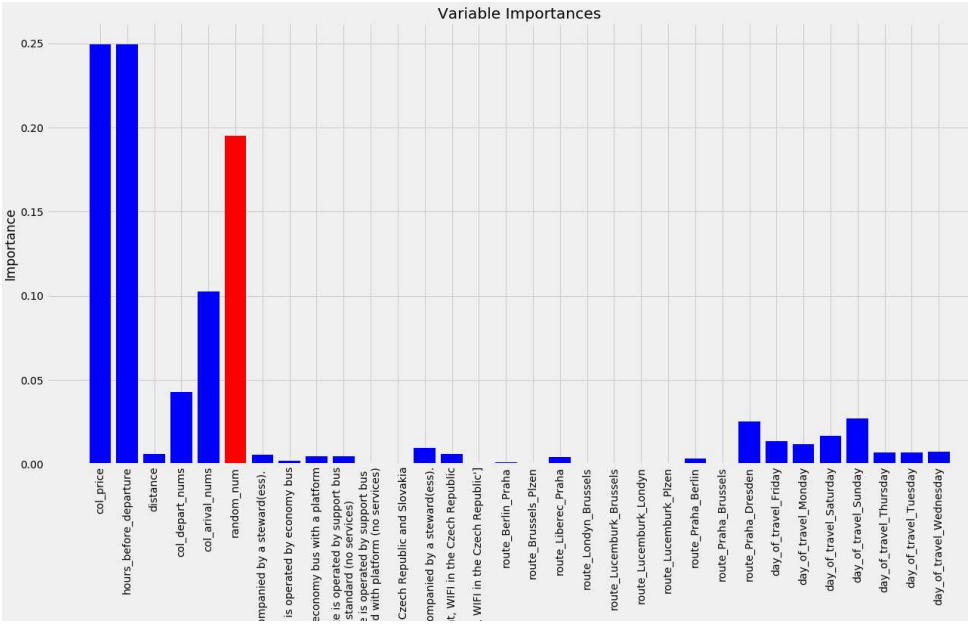


Figure 8.1: Feature importance with random variable using MDI

common comparison ground. It allows us to how see different models may attribute similar or different importance to the various variables in the dataset.

While evaluation of feature importance using MDI is widely used and a somewhat trusted method, in recent years questions have been raised regarding the robustness of using such a concept [17]. It has been shown that MDI is seemingly biased towards numerical features in the dataset and can inflate their importance. The reason this happens is because MDI is measured on the training set and given enough time during the training, the model can easily overfit on features (even on non-important ones).

To serve as a sanity check, a small experiment was conducted in which a randomly generated numerical column was added to the dataset. This randomly generated column is totally unrelated to the target variable so we would expect it to receive a relatively low or close to zero importance score.

But what actually happens as can be observed in Figure 8.1 the random_num variable (red bin) added becomes the third most important variable clearly showing the misleading nature of MDI. To further confirm this we present the evaluation metrics in Table 8.3 where the disparity between training and testing metrics is quite clear and indicates the model has overfitted to the training data and hence, performing relatively poor on the test data. Thereby, also showing the pitfalls of use of MDI in calculating feature importance.

The alternative to MDI is the use of mean decrease in accuracy (MDA) or permutation importance defined earlier in Section 3.2 for computing feature importance. MDA driven evaluation is much more robust and to show its reliability the same experiment as with MDI was also repeated and the

Metrics	Training Set	Testing set
RMSE	4.44	9.88
R-Squared	0.96	0.83

Table 8.3: Evaluation results of random forest trained with a random variable

resulting feature importance can be observed in Figure 8.2. Here we can see the added random variable is having the lowest score (equal to zero) as should be expected, demonstrating the reliability of MDA and we can now further work with the results.

We used MDA to determine the three most important features for our price and free capacity models for both Student Agency and FlixBus datasets which are reported in Table 8.4. Feature hours before departure is a recurring theme throughout table, coming out as the most important feature for three out of the four models. Price models are having free capacity as one of the top most important features while the reverse is happening for free capacity models where the price is one of the most important features. This is something which we would expect as two features are closely related although the relationship is quite non-linear as demonstrated by our analysis in section ref.

FlixBus Price Model	FlixBus Free Capacity Model	Student Agency Price Model	Student Agency Free Capacity Model
travel time	hours before departure	hours before departure	hours before departure
free capacity	price	free capacity	price
day of travel	travel time	departure time	departure time

Table 8.4: Top three most important features for the various models

Local Interpretation

As demonstrated in the previous section evaluating feature importance is quite useful and insightful interpretation technique which gives us an overall outlook of the model. But we are also interested in understanding the model's working on a granular level such as for one particular observation. This can be defined as the local interpretation of the model. Local interpretations help us to understand the decision trajectory followed by the model to take a decision on a particular observation.

To make these local interpretations we will be using a framework called as Shapley Additive Explanations (SHAP) [13] which applies a game theory

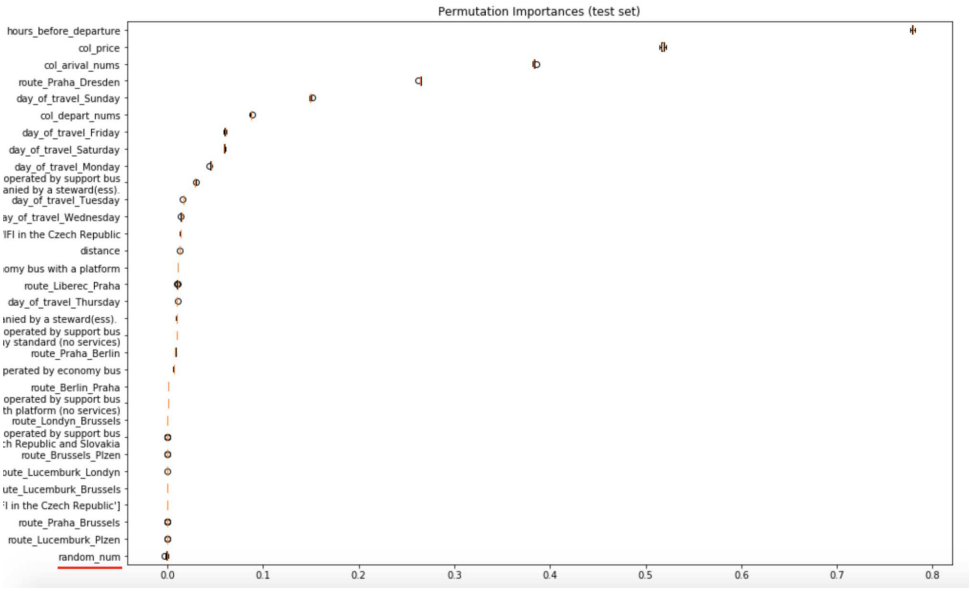


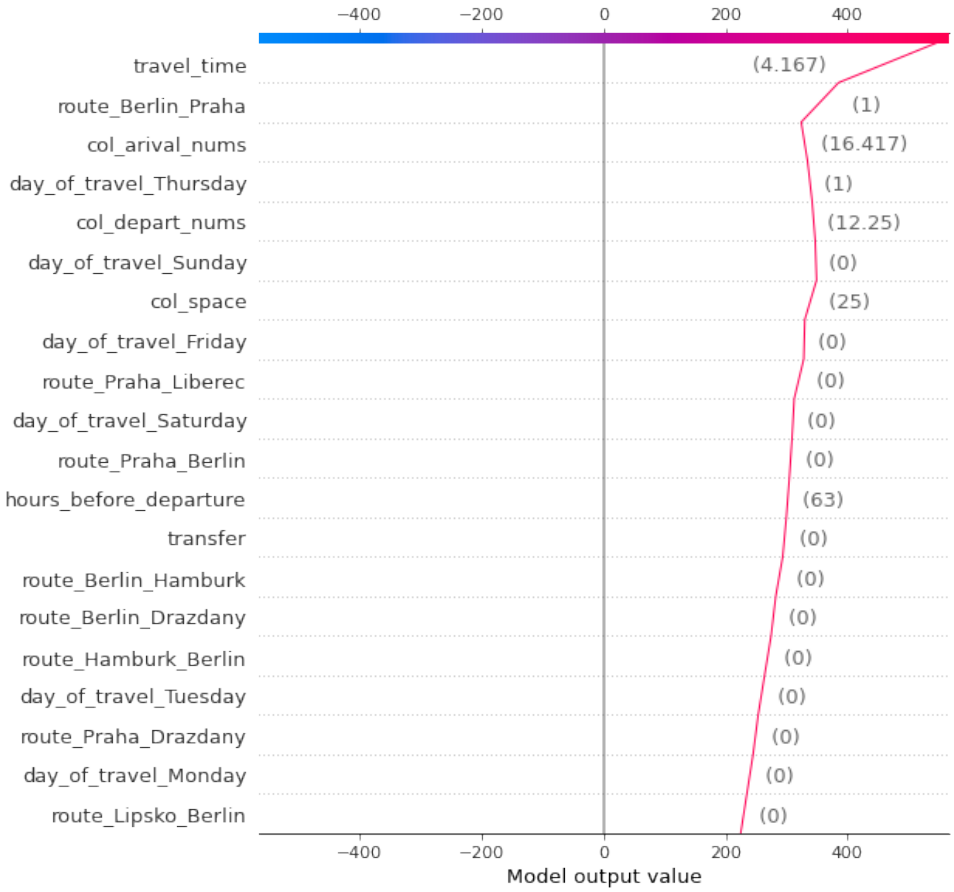
Figure 8.2: Feature importance with random variable using permutation importance

approach towards the interpretation of machine learning models. The core idea behind SHAP is to treat the model prediction as the outcome of a game where players are the features in the dataset cooperating together to bring about a particular prediction. Then each feature receives the payout or simply the contribution towards making a single prediction. We are using the Python package of SHAP to make local interpretations of our best models.

In figure 8.3 we can observe the SHAP explanation of a prediction made by FlixBus price model, where the model has predicted price of a ticket as 544.93 CZK based on the input it received while the true value is 549 CZK. We can observe in summary plot how the different features are pulling and pushing the model's output away from the base value. Base value can be understood as the average predicted output of the model. We can see the relative effect of each feature (brackets denote feature's input value) in bringing about the prediction in the detailed overview. While travel time, route and free capacity features are trying to drive the model's output towards a higher value there efforts are countered by the features arrival time and day of travel. It is also quite interesting to see how the value of each feature is affecting the whole prediction, the fact that the model is seeing that the day of travel is Thursday that helps it lower to the predicted output.

Another explanation is given in figure 8.4 where we are using the FlixBus free capacity model. The model has predicted the value to be 31.28 free seats in the bus which is quite close to the true value of 32. In The summary plot we can observe the major features that were instrumental in pushing the model's prediction for this observation from the base value of 27.94 to 31.28. While seeing that the hours before departure are still high, the route is Dresden to Berlin and the departure time is 16:00 caused the predicted

(a) Summary



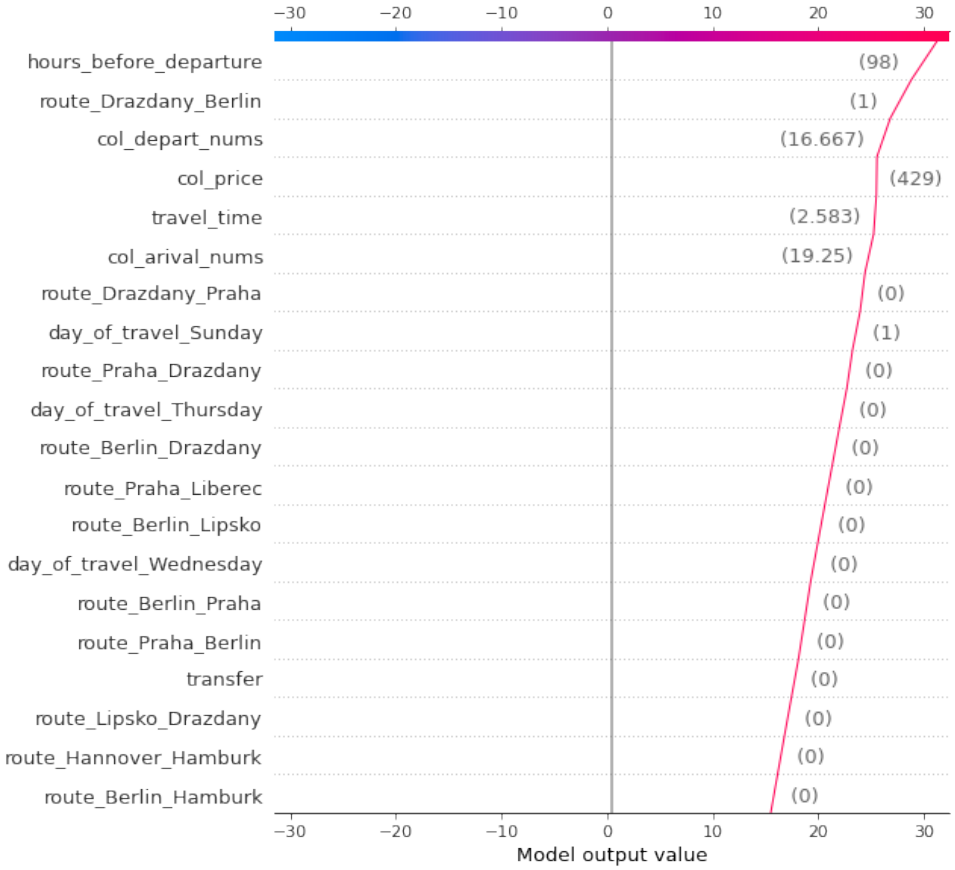
(b) Detailed overview

Figure 8.3: SHAP explanation of a prediction by the FlixBus XGBoost price model

output to be pushed further from the base value. On the other hand this was controlled by the price of the ticket and travel time and other features ultimately helping the model to settle down at the value it predicted.

These were few examples of how we can understand the working of our models on a local level. In the last section we discovered the most important features overall for each model and now with local interpretations we saw how different features and there different values contribute towards making a prediction. It was also interesting to see how certain features which might not

(a) Summary



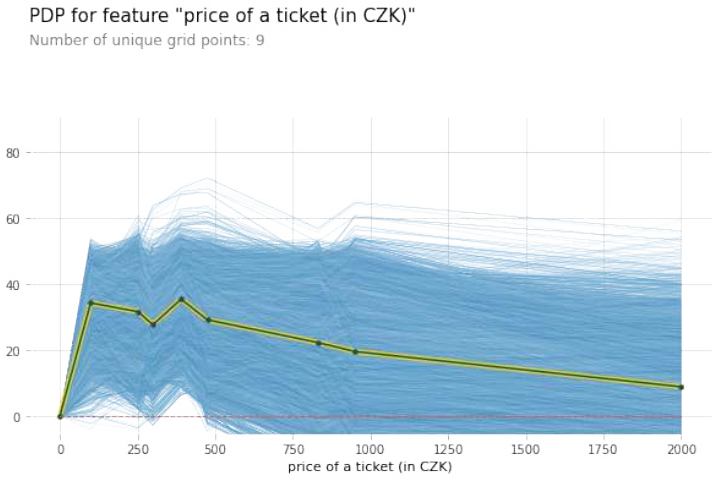
(b) Detailed overview

Figure 8.4: SHAP explanation of a prediction by the FlixBus XGBoost free capacity model

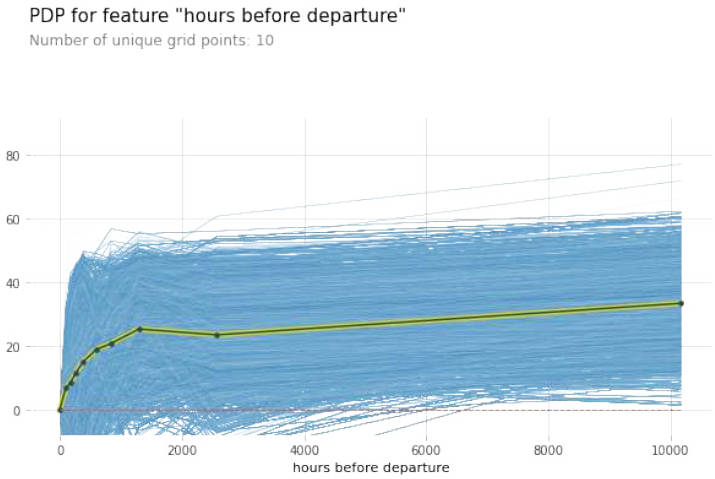
seem significant on a global level, can be observed being very instrumental for local predictions. This was was the case where we saw a huge effect from the route feature in figure 8.4 towards making that particular prediction whereas, the feature overall though is much lower in global feature importance ranking for the model.

Analysing the Partial Dependence and observed pricing policy

Final interpretation technique we will use is partial dependence plots. Partial dependence plots (PDP) allow us to investigate the relationship between



(a) Relationship of free capacity with price



(b) Relationship of free capacity with hours before departure

Figure 8.5: Partial dependence plots with the two most important features for the Student Agency free capacity model

the target variable and the features by isolating one feature at a time and evaluating it. This is similar to what we did earlier in Section 6.2 but here the key difference is the output is not from the true values but rather from the predicted value of best XGBoost models. Partial dependence plots can help us to understand the relationship much better as compared to what is observed in Section 6.2.

Figure 8.5 shows the partial dependence plots generated for the two most important features (as per evaluation in the previous sections) in the Student dataset i.e. price of the ticket and hours before departure of the bus. The plot

was obtained by fixing the value of the independent feature and calculating the target variable based on that. In other words the price of the ticket was fixed to for example 500 and then all the rows in the datasets were taken where the price equals 500 and the free capacity of the bus was calculated which is represented by the multiple blue lines. This process is repeatedly carried out until we evaluate all available values of price in the dataset and the in the end average trend is represented by the dark black line surrounded by a slightly thick yellow boundary.

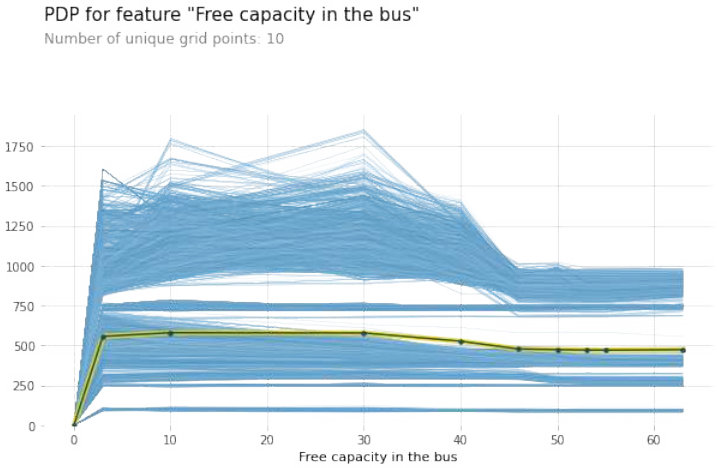
Interestingly we can see there is some linear behaviour shown in the relationship of price with free capacity in the range of price up to 150. This could be largely due to the routes such as Prague - Liberec where price typically has only 3 levels (79, 89 and 99) and the effect on price is quite small when related to the capacity of the bus. But as we look further we can easily observe the price rising with decreasing free space in the bus so everything seems to be in order here.

Similarly to price, a partial dependence plot was also obtained for hours before departure by fixing its value to the various features in the dataset. On analysing the average curve we can observe the relational is almost hyperbolic showing that the fluctuations in the capacity of the bus do not start to occur until 2000 hours (approximately 3 months) before the departure of the bus. This again goes in line with known consumer behaviour of not indulging in booking tickets especially for buses as they have relaxed cancellation policies and also quite possibly because the bus travel can be planned quite instantaneously.

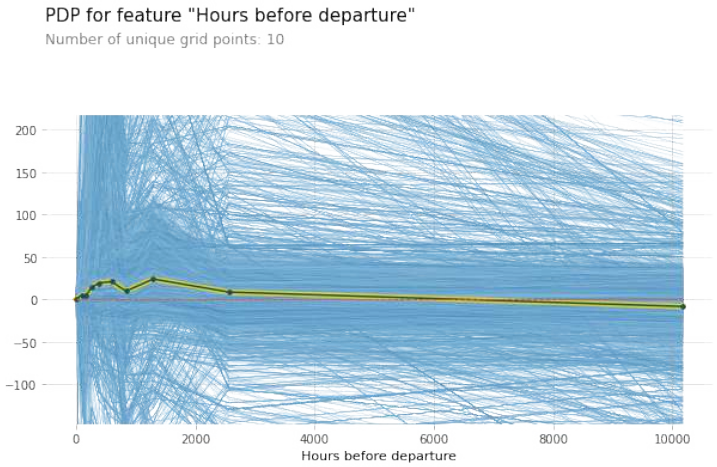
For FlixBus price model we discovered the top two features were travel time and hours free capacity of the bus in Table 8.4. The relationship of price is shown in the PDP plot in Figure 8.7. We can see a very similar hyperbolic relationship of the average predicted price with free capacity as we saw previously with the Student Agency price model. Here again the price is rising as the bus is filling up showing the tendency of bus service providers to increase ticket prices when the bus is more occupied.

Student agency is using dynamic pricing which seems to be heavily reliant on hours before departure of its buses and free capacity in the bus along with the departure times of their buses and we can observed how these relationship look like in Figure 8.6 and how they affect the pricing overall. For FlixBus we discovered the scenario slightly different where we observed through our interpretations the most important features in determining the price are the time taken to complete the journey and the free capacity in the bus. A strong linear dependence on travel time combined with available free capacity seems to be the biggest contributors towards FlixBus's dynamic pricing policy.

Having discovered the most important features globally along with some local interpretations and PDP plots, this now brings us to end of the journey we started off by exploratory analysis of features affecting the pricing of tickets in Section 6.2. There we received just an early gist of what is happening and how relationships might look like. Our interpretations of the best trained models have further allowed us to validate our beliefs (existence of highly non-



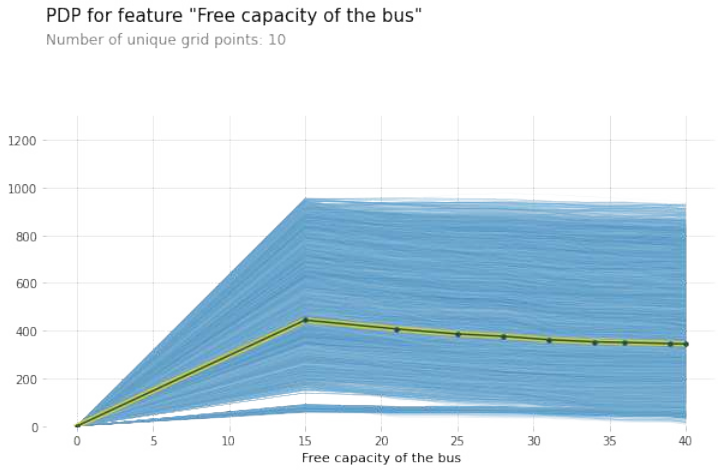
(a) Relationship of price with free capacity in the bus



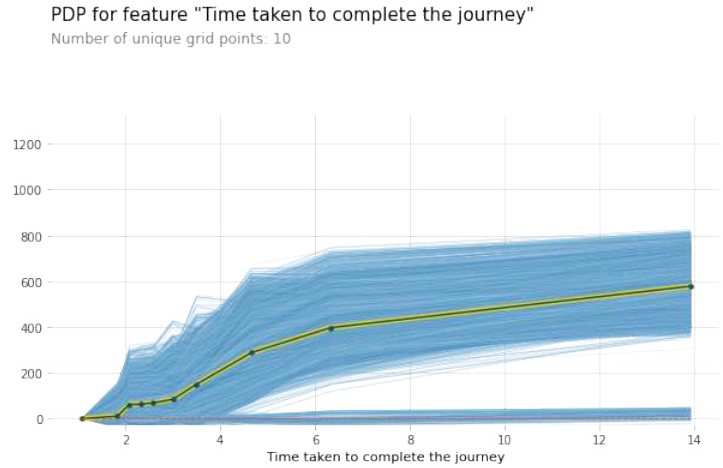
(b) Relationship of price with hours before departure of the bus

Figure 8.6: Partial dependence plots with the two most important features for the Student Agency price model

linear relationships and as well redundant variables) as well new discoveries were made such as finding the most important features in the datasets along with more clearly emphasize relationships amongst the features were observed.



(a) Relationship of price with free capacity in the bus



(b) Relationship of price with time taken to complete the journey by bus (in hours)

Figure 8.7: Partial dependence plots with the two most important features for the FlixBus price model

Chapter 9

Conclusion

In the due course of our work we showed how Student Agency and Flixbus are dynamically pricing their connections based on strong influences from features such as free capacity, hours before departure of the bus and travel time.

We found out the underlying relationships between the features in the datasets to be highly non-linear through early exploratory analysis. As we used non-linear machine learning techniques such as random forest and XGBoost it allowed us to fit better to the data. Ultimately our XGBoost model outperformed other methods based on the evaluation metrics reported in Tables 8.1 and 8.2. It was able to learn well the underlying relationships of price and free capacity with the features in the datasets. It was then used further for making predictions and understanding relationships with features in the model. Based on the results of the work done we can say that we have achieved the objectives set at the beginning of this thesis to explore the pricing strategies being used and to train our machine learning models which can learn them well and use them further for predictive analysis.

The work done in the thesis can be further expanded by a more deeper analysis based on different bus connections. Using multiple smaller models which are predicting only for particular routes rather than one larger global model. This could further reveal pricing models being used on a local level by the bus service providers. Another avenue of expansion is through analysis and training of models on more recent data which could reveal whether the pricing model has remained same or it is evolving with time.

Bibliography

[1] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[2] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[4] Real Carbonneau, Kevin Laframboise, and Rustam Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154, 2008.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[6] Jan Drchal. Statistical machine learning (be4m33ssu) lecture 12: Ensembling. 2019.

[7] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[8] Alberto Gaggero, Lukas Ogrzewalla, and Branko Bubalo. Pricing of the long-distance bus service in europe: The case of flixbus. *Economics of Transportation*, 19, 09 2019.

[9] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[11] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[12] Neal Lathia and Licia Capra. Mining mobility data to minimise travellers' spending on public transport. In *Proceedings of the 17th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, KDD '11, page 1181–1189, New York, NY, USA, 2011. Association for Computing Machinery.

[13] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[14] Stephen Milborrow. Cart tree titanic survivors, 2011. CART tree titanic survivor.png.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] Joseph Rocca. Ensemble methods: bagging, boosting and stacking, 2019. https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.

[17] Christopher Csiszar Terence Parr, Kerem Turgutlu and Jeremy Howard. Beware default random forest importances, 2018. https://explained.ai/rf-importance/.

[18] Newcastle University. Residuals. https://internal.ncl.ac.uk/ask/numeracy-maths-statistics/statistics/regression-and-correlation/residuals.html.