Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Jan Zikeš**

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Data-driven job allocation in taxi services with autonomous drivers**

Guidelines:

1. Familiarise yourself with the taxi order allocation problem.
2. Identify and analyse selected available data sets about order allocation in taxi services.
3. Formally specify the order allocation problem for taxi services with autonomous drivers.
4. Design a multi-agent mechanism for taxi order allocation.
5. Implement the designed mechanism.
6. Evaluate the properties of the mechanism on a testbed based on real-world data.

Bibliography/Sources:

[1] K. Seow, N. Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," IEEE Transactions on Automation Science and Engineering, vol. 7, July 2010.
[2] R. Bai, J. Li, J. Atkin, and G. Kendell, "A novel approach to independent taxi scheduling problem based on stable matching," Journal of the Operational Research Society, 2013.
[3] A. Glaschenko, A. Ivaschenko, G. Rzevski, and P. Skobelev, "Multi-agent real time scheduling system for taxi companies," in Proc. of 8th Internation Conference on Autonomous Agents and Multiagent Systems, 2009.

Diploma Thesis Supervisor: Ing. Michal Jakob, Ph.D.

Valid until the end of the summer semester of academic year 2015/2016

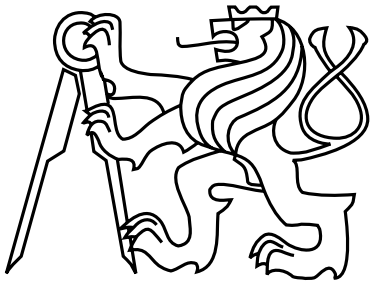
L.S.

doc. Ing. Filip Železný, Ph.D.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, October 31, 2014

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science



Master's Thesis

Data-driven job allocation in taxi services with autonomous drivers

Jan Zikeš

Supervisor: Ing. Michal Jakob, Ph.D.

Study Programme: Open Informatics

Field of Study: Artificial Intelligence

January 4, 2015

Aknowledgements

I would like to thank to the supervisor of this thesis Ing.Michal Jakob, Ph.D, who has helped to the creation of this work by his guidance, encouragement and his advices through the whole process of writing and the development of the Master's Thesis.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on January 4, 2015 .....

Abstract

In our work we have first formalized the matchmaking mechanism that is currently used as a part of the novel taxi booking system using smart phone applications. Then we have identified the system that is selecting the drivers to who the request should be sent as a critical place for the improvement of the existing mechanism.

Then we have formalized the sub-problem of selecting the particular most relevant drivers to who the request should be sent. After this formalization we have proposed the probabilistic classifier model as a one from possible and also very natural solution of how to improve the matchmaking mechanism.

Afterwards we have also analyzed the dataset that was available to us. We have also performed several visualizations of the data. Then we have implemented and experimented with several additional feature extraction methods from the available dataset.

In the next section of our work we have implemented several scripts in Python that easily enabled us to learn various kinds of models from the available data. We have particularly learned Naive Bayes model, K nearest neighbors and decision tree forest model. For all the mentioned models we have experimented with various features from those that were available directly form the data to those that we had to artificially derive and compute based on the various fields in the dataset.

At the end we have performed evaluation of all the mentioned models. We have first performed evaluations of the models on its own and then also as a integral part of the matchmaking mechanism. We have also come up with several recommendations towards the implementation of our proposed models into the real world production system of our industrial partner.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Transportation in the cities has always been fascinating mechanism, together with the fact that there are still more and more people moving to the cities almost all around the world[1]. This fact demands still more and more innovations to keep the growth of the transportation systems in the cities sustainable. One possible way, how the sustainability can be reached is utilizing emerging new technologies that can help us to organize transportation in a smart way and thus make it more accessible and convenient for all the inhabitants of the cities. The modern technology can either help us to make traveling more comfortable or it can also gives us the ability to collect and analyze the data, which can be then used for the further analysis and optimization of the transportation.

1.1 AI and modern technology in transportation

In the recent years we could have also noticed that there has appeared a huge amount of innovations in the fields of transportation on demand, ride sharing and also particularly in the taxi industry. All around the world there were introduced various services using advantages of the modern technology, such as smart phones or also other "smart" devices. These modern technologies has in many places of the world almost fully replaced the from history well known cab hailing or dialing the dispatching service phone number to order a taxi. From pioneers of connecting the technology and taxi industry in the world we can name famous companies such as Uber, Lyft, Hailo or mytaxi. From the Czech Republic we should mention liftago that is starting to be very active in implementing also various new innovations, but there are also other more or less successful projects usually owned by already existing taxi dispatching companies.

On the other hand there was recently performed also huge amount of work in the field of various data collection and analytics based on the data. We can maybe say that we are experiencing some kind of hype around fields such as machine learning, statistics and also in other related fields. In recent years we could have also seen the whole newly created or at least named fields that were basically based on previously known principles from mentioned machine learning and statistics, but it also incorporates the other fields. From fields that

[1] <http://en.wikipedia.org/wiki/Overpopulation>

has recently appeared we can name for example Big Data, Data Science or Internet of Things.

But when we are speaking about Data Science or Internet of Things we can notice that certainly the previously mentioned modern transport on demand and taxi booking systems are not just simplifying and speeding up the way we order the transportation, but these systems also has potential of generating huge amounts of the data from various sensors and devices. These devices can be modern technologies such as smart phones, GPS devices, but also some old improved devices such as taximeters. All the recorded data can by possibly analyzed for either further improving the own business of the industry innovating companies. This can also lead to decreasing of the fuel consumption and decreasing of the emissions produced by the taxis, but we can also imagine the data usage, together with the growth of smart devices penetration among the city inhabitants, even for some global improvements of the whole transportation systems in the cities such as traffic control systems or dynamically setting the price of road tolls.

As one of the the first companies that are trying to connect applications for booking the transportation on demand or taxis with the data analytics we can name Uber which data department also sometimes publishes some interesting insights on their blog[2]. But also other companies are starting to invest to the connecting of the industry with data analytics for example also the previously mentioned Czech liftago.

We can also imagine that the mobile application for the transport on demand or taxi ordering together with the data analytic might be just the beginning. For example upcoming era of self driving cars might even help us in the analyzing the data by removing non optimal and not fully deterministic behavior that we can sometimes experience on the side of the taxi drivers. Future mechanisms can also combine data analytic with coordination and resource allocating mechanisms same as also combine the taxi booking systems with other modes of transportation such as shared bikes or other standard types public transport.

1.2 Vehicle passenger matchmaking mechanisms

One from the most important problems that is needed to be solved by the vast majority of the above mentioned companies is passengers matchmaking mechanism design on which we have also focused in this work. This problem basically deals with the allocation of the transportation resources such as taxis or other kinds of vehicles on demand to the passengers. In this problem we have especially paid attention on the optimality of the allocation on the both sides and on the availability of the transportation resources to the passengers.

Since this problem has two sides it is usually difficult to reach some optimal situation for both sides the passengers and the drivers, but we can try to perform various optimizations to connect passengers with the most optimal vehicles for them from those that are available in the city.

[2] <https://blog.uber.com/tag/uberdata/>

1.3 Approach to the problem

In our work we have particularly focused on the improvement of the already existing novel technology using systems. We are especially interested in the optimization of the vehicle passenger matchmaking mechanisms. Thank to the fact that we have available data from the real world system operation we are able to use advantage of it and study various machine learning methods that can be used thanks to the available logged data about the history of the system service. The data was provided to us by the leading Czech taxi industry innovating company that wish not to be named in the work. In this work we have tried several machine learning methods and we have also proposed it's evaluation framework that can help us to compare all the similar methods to the our proposed one.

1.4 Objectives of this work

Our goal in this work is to introduce and experiment with several methods that would connect modern taxi allocating companies systems with the data analytic mechanisms mostly known from the field of machine learning and improve it's taxi allocation efficiency. This might help us to move current state of the taxi allocation systems few steps forward towards the optimal self organizing autonomous systems that we can imagine to surround us in the future.

Our second goal is to show that there it is really still possible to optimize currently working vehicle passenger matchmaking mechanisms and to improve these mechanisms in a way that it will be also possible to use them autonomously in the future era of self driving cars.

Chapter 2

Related Work

Because of this work is touching not only optimization of the transportation resources allocation, but also the data mining and machine learning we have to divide this section to three different parts. From which one is dealing with the transportation itself and different possible approaches of how to optimize it, the second part is dealing with usage of the coordination mechanisms and also about it's usage particularly in the transportation domain and finally the third part that is introducing the related work in the field of machine learning, especially methods that can be used to improve the transportation on demand or taxi allocation.

2.1 Transportation on demand or taxi allocation

Generally topic of our work can be classified as the optimization in the field of transportation. For our work was the main focus was on fields where we can do some improvement by modern computing techniques such as artificial intelligence, machine learning, multi-agent systems or any other related approaches.

We went through many different papers dealing with the usage of the publicly available data and existing transport related real time APIs. For example there is interesting research paper dealing with the traffic network analysis in London [1] or another paper dealing with the real time traffic modeling and estimation [2]. We also went through several ideas and futuristic papers describing what research can possibly be done in the future such as [3].

2.1.1 Towards taxi system optimization

Thank to the dataset that was provided to us we have decided to focus especially on the taxi booking mechanisms and to it's optimization based on the data. This domain was previously also investigated, particularly there is probably being made serious research by the US companies such as Uber or Lyft, at least based on how they present themselves. Unfortunately there is publicly available probably only a fraction of the research that is being made by mentioned companies. On the other hand there has previously has been performed some research in the academia that is most of the time publicly available.

2.2 Transport resource allocation

First possibility of how to treat transportation optimization problem is as a multi agent coordination problem. Here we have especially focused on the limited resource allocation systems of the various kinds from the shared bicycles to the optimal taxi allocation mechanisms.

2.2.0.1 Parking lots allocation optimization

There has recently been proposed several papers dealing with the parking optimization. First of all there is couple of papers describing how the system should be designed from the software engineering point of view, for example [8] or [9].

Then there was made some work also in the field of usage of multi-agent systems in order to optimize parking lots allocation. In the majority of papers there are used methods such as multi-agent negotiation in [10] or slightly more complex approach in [11].

There was also found some literature on global optimization approaches to solve the problem. The most interesting is [12] where was used tabu search in order to optimize the parking lot allocation. Unfortunately this approach is feasible only on very small instances of the problem.

There is also worth of mention that there already exists successful implementation of the intelligent parking system in San Francisco[1].

2.2.0.2 Shared vehicle allocation optimization

There is also some related literature that is dealing directly with bike, car or other vehicle sharing. Majority of the related literature is dealing with the analysis of current working bike sharing services. For example analysis and system expansion recommendations in the Washington D.C. metropolitan area [13]. Then another paper that analyzes bike sharing systems in various cities [14] mainly from the demand and its distribution point of view. And again very similar analysis was performed on the bike sharing system in Vienna in [15].

There is following some work where authors are trying to compute the best possible locations where the bike sharing stations should be placed. In the wast majority of cases there are used multi-criteria optimization methods. For example in [16] are authors trying to optimally set up the network in Athens. Or in [17] authors has used genetic programming to optimally position the stations in Snatander.

Much deeper analysis of the bike sharing system in Lyon was performed in [18]. In this work there was for example detected which stations are wrongly positioned, what are the passenger behavior patterns and there was also introduced some basic system, how to do some predictions from the data.

[1] Intelligent parking system in San Francisco <http://sfpark.org/how-it-works/>

Step further was made in [19] where authors have proposed and analyzed two systems for the re balancing of the bike sharing stations. First approach was optimization of the vehicles that are transporting bikes from full stations to the empty ones. Second approach was some basic proposal of dynamic pricing.

There is also some work dealing with the optimization of the car sharing systems with pre-booking needed from the side of the passengers. In [20] they have proposed multi-agent system with in advance reservations and negotiations.
There is also first related work [21] where authors performed analysis of the current bike sharing system in Singapore using the stochastic network flow model.

2.2.1 Taxi systems optimization

Particularly in the taxi industry there was made some work that deals with the taxi dispatching system optimization from the multi agent systems point of view in [22] or in [23]. There has also been done some previous work in the mechanisms for taxi allocation in the work related to the CTU in [24].

2.3 Transportation optimization using machine learning

There has already been done some work dealing with usage of the machine learning methods in the domain of transportation, but on the other hand there was found only a few papers that are particularly dealing with the taxi allocation mechanism using machine learning methods.

2.3.1 Learning approaches to forecast the traffic

Work in this field seems to be very inspired by the machine learning methods dealing with internet traffic flow. Here the interesting overview of the work that has previously been done can be found in [25]. Then there was found several papers dealing with traffic prediction, the most interesting were approaches where the authors are using Bayesian networks approach [26] and time series analysis approach [27].

2.3.2 Learning approaches in the taxi dispatching domain

Particularly in the taxi dispatching domain we can divide the work that has been made to two groups. In the first group there are authors trying to estimate how will the upcoming request look like. For example where will be the place to pickup the next passenger, or what will be his most probable next passenger's destination. Interesting blog post about estimating this was written by the uber data team[1]. Overview on passenger hunting strategies based on the machine learning methods was previously studied in [28] and in [29]. There were also performed some experiments with mechanisms predicting taxi passenger demand in [30].

[1] <https://blog.uber.com/passenger-destinations>

Second group of works is looking to the taxi allocation problem from the other side. It is trying to predict for passengers where they can find vacant taxis. Which is viable problem especially in a big cities such as New York or London, where getting a taxi in certain time of the day might be a serious problem. This problem is widely described in [31]. There was also found research paper [32] that is trying to combine the both already introduced approaches to the taxi system optimization.

2.3.3 Learning methods using taxi companies data

Another interesting research area is usage of the taxi companies data for learning other interesting patterns in the transportation. Here we can mention especially [33] where authors are trying to learn the interesting areas in the cities based on the time of the day and it's corresponding movement patterns from the taxi history GPS data. In the second interesting paper [34] there has authors assumed that taxi drivers are very skilled in terms of city navigation and thus they have used the data from the taxi drivers to learn the fastest routes in the city.

2.4 Related work summary

We have found quiet big amount of related literature that is dealing with the transport resource allocation in general, we have also found some literature that is starting to use machine learning methods for various transportation optimization. On the other hand there was found only a few papers that were dealing with some kind of optimization in the taxi allocation in particular. There was also found no literature that was trying to combine the machine learning with taxi passenger matchmaking mechanisms, this provides the opportunity for our work to to perform a research that has already not been described in the literature.

Chapter 3

Matchmaking mechanism

In this section we will describe the whole matchmaking mechanism from global point of view. We will start with the description of how we model the protocol of the existing taxi booking mechanism and how we formalize the whole matchmaking mechanism. Then we will introduce the sub problem called the request recipient selection problem, which is the most important sub problem for us. Particularly in this section we will deal with the formalization of the problem and with the way of how we can evaluate the request recipient selection problem. On the other hand in the following section 4 we will describe one possible way of how to solve the request recipient problem. At the end of this section we will describe in detail all the datasets that we have available for this work in order to optimize the matchmaking mechanism.

3.1 The taxi ordering protocol

The goal of our work is to optimize the currently existing mechanism of our industry partner that has the following structure: There are two mobile applications and one server side application. One mobile application that is dedicated to the passengers and one that is dedicated to the taxi drivers. The server side basically only mediates the communication between both passenger's and driver's mobile applications. For better understanding of the whole taxi ordering protocol, it can be seen in the UML sequence diagram 3.1.

You can pretty well notice that there are different possible actions at a certain point of the communication on both sides of the protocol. First of all we can obtain three different reactions from the taxi driver to the passengers request. It can be rejection, timeout or acceptance. Because of in the real world we are mainly interested only in accepts we have decided to treat both reject and timeout like the same class called reject. For as accurate as possible modeling of this step we have decided to introduce the driver response model in the following section 4.

On the other side of the system there can each of the passengers decide if he would like to accept some offer from the driver or if he decides not to select any of the offers. Again for modeling of this step we have decided to later introduce the so called passenger's model which is very similar to the already mentioned driver response model.

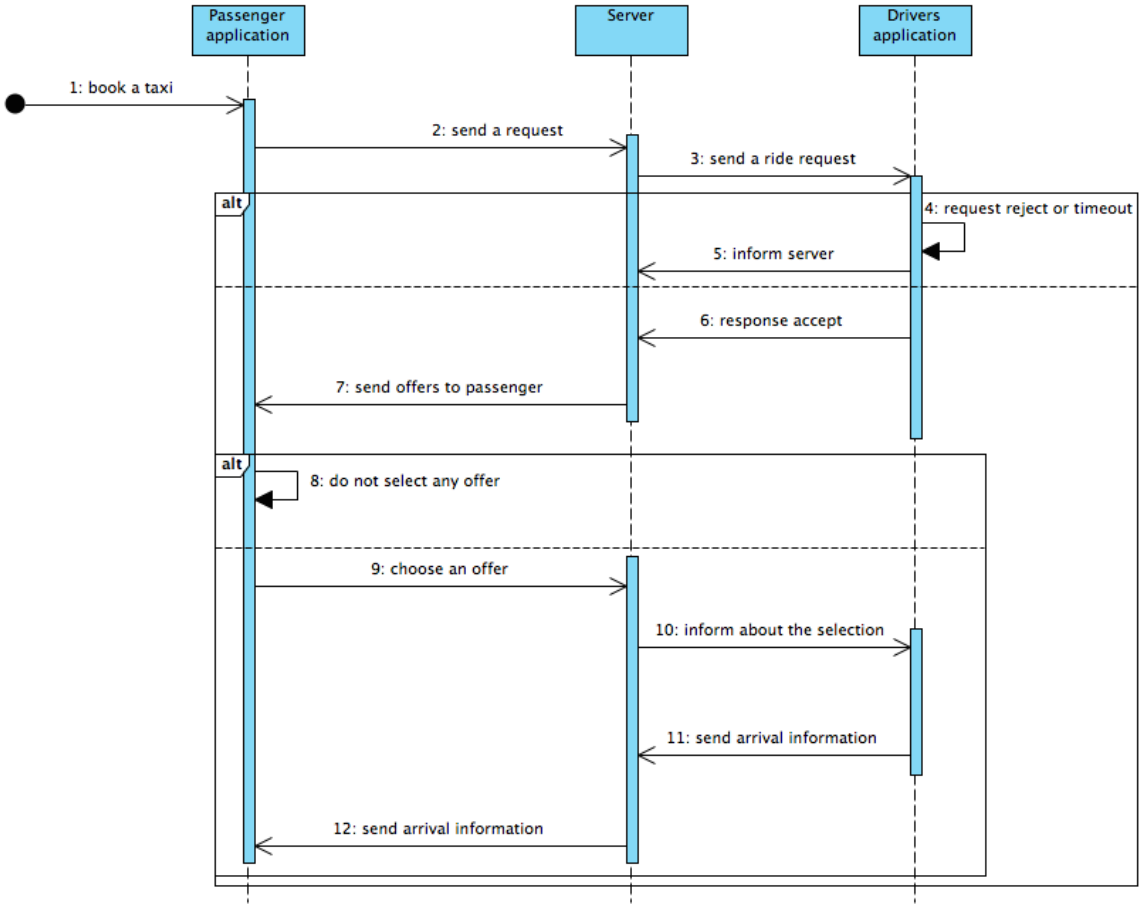


Figure 3.1: Sequence diagram of the taxi booking protocol

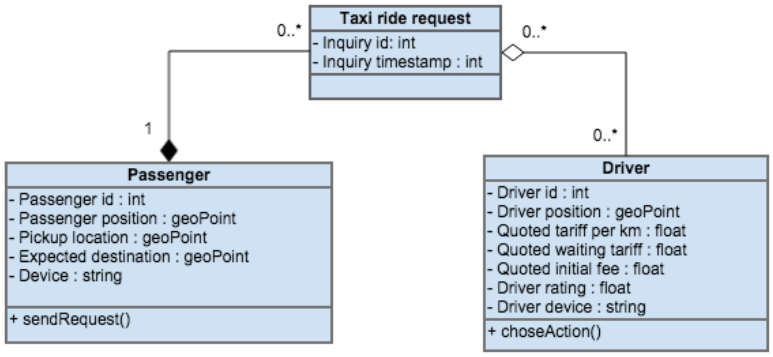


Figure 3.2: Class diagram of the data structures used by matchmaking algorithm

3.1.1 Matchmaking algorithm inputs

First of all let us describe what are our known inputs of the matchmaking algorithm. We have tried to sort them from the high level to the low level inputs. For better understanding of the matchmaking algorithm from the data point of view there was also presented the UML class diagram which describes more in details the entire data flow. The diagram can be seen in the figure 3.2.

3.1.1.1 Map - graph of the city

In our work we are dealing with the geographical data on the map of the city. The city map is represented as a Graph $G = (V, E)$ where edges E are representing the connections between intersections on the map. Every edge is a organized two-tuple $E = (d, t)$, where d is the edge length and t is the time that is needed to travel along the edge. And vertices V are representing the intersections.

3.1.1.2 Taxi drivers

One side of our system consists of the list of taxi drivers each in detail described by driver descriptor shown in the figure 3.2. Each driver has Driver ID which is the unique identifier of the driver in the system, Driver position which is the driver's actual position, Quoted tariff per km that is driver's actual tariff per km of the ride with passenger, Quoted waiting tariff that represents the driver's price per one minute of waiting, Quoted initial fee which is the fee that every passenger has to pay to the driver when he enters the taxi, diver's rating represents the actual driver's rating in the system which is a real number from 1 to 5 and Driver device represents the type of the mobile device that the particular drive uses.

3.1.1.3 Passengers

On the other side of our system we have the list of passengers. Same as in the case of taxi drivers even for the passengers we can see each passenger described by the passenger descriptor in the figure 3.2. Each passenger has Passenger ID which is the unique identifier

of the passenger in the system, Passenger position which represents the passenger's actual geographical position, Pickup location that represents the geographical point of the desired journey origin, Expected destination which represents the geographical point of the desired passenger's destination and Device that represents the type of the device that on which the passenger has installed the application.

3.1.1.4 Taxi ride request

Passengers are then sending requests to the mediator server. Each request contains the information about at least one passenger. The mediating server side system then chooses all the relevant taxi drivers to who the passenger's request should be passed. From this information we are getting to the resulting list of requests that contains records where each consists of both all the information about the passenger and all the information about the driver. In case that there are no drivers available in the system then there is created no request and the empty request list is sent back to the passenger. In addition to the passenger's information and information about all the selected drivers the taxi ride request contains also Inquiry id which contains the id of the taxi ride request and inquiry timestamp that represents the time when the request was created. Again we can see the taxi ride request description in the figure 3.2

3.1.2 Matchmaking algorithm outputs

The desired output of our matchmaking algorithm is the particular allocation of the passenger to the particular taxi drivers.

3.2 Request recipient selection problem

From the description of the whole matchmaking mechanism we could have noticed that the matchmaking mechanism has several steps. First of all there have to be sent a taxi ride request by the passenger. Afterwards there have to be performed some kind of selection of the recipients among taxi drivers and at the end there will the passenger perform his selection of the taxi driver. Particularly the request recipient selection problem is a sub problem of the matchmaking mechanism that contains all the mentioned steps except the selection of the best driver that is performed by the passenger.

The main reason why we have particularly focused on the improving of the mechanism by solving and optimizing the request recipient selection was the demand from our industry partner for who it seemed to be one from the most problematic part of the currently working mechanism. From the data that we have available we can also see that in this part is really big potential for the improvement of the matchmaking mechanism. From the comparison of the newly created dataset from the activity data and the transaction dataset we could have also noticed that transaction data contains only a fraction of the drivers that were available for every transaction. Thus we can actually say that there was already performed some kind of request recipient selection. On the other hand here comes the question how optimal and by which mechanism were performed the particular selections that were made and that we

can see in the transaction data. Based on the conversation with the company providing us the data there were used some kinds of algorithms using the heuristic knowledge. The goal of our work is to improve this request recipient selection algorithm using the heuristic knowledge, by some our more general on machine learning based approach.

3.2.1 Inputs of the request recipients selecting algorithm

As the input to our request recipient selection algorithm we are basically getting only the passenger's request for the particular ride and on the other side the database of all the available taxi drivers at the certain moment. This data structure can be formalized as the list of taxi ride requests as it was described in the section 3.1.1.4 where requests differs one from each other only in it's driver part.

3.2.2 Outputs of the request recipients selecting algorithm

The desired output provided by the request recipients selection algorithm is the list containing N selected taxi drivers as they were described in the section 3.1.1.2. We can notice here that only remaining step to solve the whole matchmaking mechanism using the request recipient selection problem solving algorithm is to select the best driver among the output which we will for our purposes let on the user of the application.

3.2.3 Request recipients selecting algorithm description

The final solution that should be provided by the mechanism can be basically divided into two different parts. First part is estimating the probability that driver will accept the passenger's request. Second part is the assignment of the N best taxi drivers for every passenger's request. Dividing the potentially only one part to the two separate parts is very much driven by the industry partner. Normally we could only broadcast the drivers with probability higher than for example 0.5, but on the other hand from the industry partner point of view there is a pressure on having at least some request with accept taxi drivers action so we better compute probabilities for all the available drivers and then we perform the selection of the N best taxi drivers.

3.2.3.1 N best taxi drivers

As the result of the first and second part of the mechanism is the selection of the N best taxi drivers from all the taxi ride requests sent to the taxi drivers in the given time. N best taxi drivers will be selected based on the probability p_{dr_a} that the driver d will accept request r. In the other words we will try to maximize $\sum_{d \in N} p_{dr_a}$. The other problem is determining the number N, for simplicity and possibility to compare the results we are later in this work using the same N as it was used in the test data, but for the implementation in the real world there still can be performed some optimizations and optimal N estimation.

3.3 Mechanism evaluation

First of all we have identified several key performance indicators that we would like to optimize in our work. Then we have proposed evaluation framework which we have used for all the evaluations of our whole matchmaking mechanism. Particularly the only change that we have focused on the matchmaking mechanism was the request recipient selection so the evaluation mainly deals with the evaluation of the request recipient selection sub-problem described in the section 3.2.

3.3.1 Key performance indicators

First from our goals is to decrease the total number of "spam" requests that are being sent to the taxi drivers. In other words we would like to increase the ratio $r_r = \frac{r_a}{r_t}$, where r_a is number of requests that was sent and that was accepted by the taxi driver and r_t is number of requests that was sent.

Second key performance indicator that we would like to optimize is the total number of accepted requests we would particularly like to increase this number. In other words we would like to increase the r_a accepted requests from the previous section.

Unfortunately in the current taxi ordering system, there is sometimes happening that for some passengers there is accepted no offer from the taxi drivers at all. So the another goal of our system is to increase number of passengers that are getting at least one offer from the available taxi drivers.

3.3.2 Evaluation framework

In order to evaluate our mechanism we have decided to introduce one more evaluation that will not evaluate only the model, but also the whole matchmaking mechanism. Particularly in the evaluation we are evaluating the proposed algorithms that are solving the request recipient selection problem as it was described in the section 3.2.

In this part we have especially focused on the evaluating predicted outcomes based on the real world data. The majority of the evaluations were made according to the following table.

		Real world		
		Reject	Accept	Don't know
	Reject			
Our model	Accept			
	RnS			
	AnS			

First of all we will try to explain this very important table more in details. The table deals with outcomes of the different passenger's journey requests. There will be recorded real world outcomes that will be available in the transaction data in the columns and the outcomes

predicted by our mechanism will be recorded in the rows. During the evaluation section there will be presented this table several times. We will try to show there both, the total counts and relative counts. Usually absolute counts as a given outcome action counts and relative counts usually in percents from all the records in the table.

3.3.2.1 Real world outcomes

Real world outcomes or in other words the the request recipient selection that was performed in the reality by the currently deployed system by our industrial partner are represented by the columns in the table. First of all you can notice the last column which is called "Don't know". In this column will be recorded all the records for which we don't know what would be the outcome, because we do not have recorded these driver in the real world transaction data described in the section 3.4.1. On the other hand in the first column we have records that has ended in the real world either by the reject action of the driver or by the driver letting the order to be time outed. And in the second column there are all the records in which we have drivers accept as the final outcome in the data.

3.3.2.2 Our system predictions

Request recipient selections performed by our system are represented in the rows. We can notice four different types of the records and thus four different outcomes. First of all we have there "reject" outcome. In this row will end up all the selected requests as it was described in 3.2.2 where we have predicted reject outcome, but still several one from the best from all the rejects to just provide certain number of the drivers that needs to be provided for the passenger. Then we have row with "accept" this row represents all the requests that has been predicted accept outcome by our model and the accept was so good that we have actually select the driver among n best drivers. Another line denoted "RnS" would contain all the drivers for which has our model predicted reject outcome and we have not selected them in the n best drivers and finally "AnS" is the group of the drivers where has our model predicted accept action, but we had already selected enough even better accepts, thus we have not selected the particular driver.

3.4 Available datasets description

For our work we have available two different datasets. The first dataset basically contains the transactions data or in other words the data that are describing the communication between passengers and drivers. The second dataset describes which drivers were active in the particular time of the day.

3.4.1 Transactions data

Our transaction data describes the communication between passengers and the taxi drivers for August and September 2014. We can look at our transactional data in two different ways. First of all we have several consecutive records that belongs to one transaction, in other words one request from the passenger is described in several records of the dataset,

because the communication with each of the drivers that was broadcasted by the system is recorded as a separate record. On the other hand for purposes of this work we can have a look at the data in a simplified way that each row of the data is independent request of the passenger to the driver.

Transactions data contains following fields[1]:

Inq_ID
ID of the transaction.

Broad_Driv_ID
ID of the driver that was broadcasted by the passengers request.

Driv_Pos_At_Inq
Position of the driver at the time of the passengers inquiry in the format latitude/longitude.

Inq_At
Timestamp of the time when the transaction has started.

Broad_At
Timestamp of the time when the transaction was broadcasted to the driver.

Pass_ID
ID of the passenger.

Pass_Pos_At_Ord
Position of the passenger at the time of the passengers inquiry in the format latitude/longitude.

Req_Pick_Loc
Passenger's desired pick up location in the format latitude/longitude.

Ride_Exp_Dest
Position of the passenger destination at the time of the passengers inquiry in the format latitude/longitude. It is available only when passenger has entered the information.

Broad_Cnt
Number of taxi drivers that were broadcasted in the particular transaction.

Driv_Act
Action that was performed by the driver, one from TIMEOUT, ACCEPT, DECLINED. For purposes of this work we have decided to treat TIMEOUT and DECLINED as the same outcome, because we are only interested in ACCEPTs.

Driv_Pr
Total amount charged by the driver at the end of journey.

[1] There were omitted the data describing the finished journey as actual driver waiting time or actual driver arrival time.

Driv_Ac_At
Timestamp of the time when driver performed ACCEPT or DECLINED actions.

Quot_Tar_Per_Km
Unit price per km in Czech crowns.

Driv_Wait
Unit price per minute of drivers waiting for the passenger time in Czech crowns.

Driv_Init_Fee
Unit price that has to be paid at the begging of the journey in Czech crowns.

Driv_Rat
Average rating of the driver, float numbers from 0 to 5.

Comp_State
The final state of the journey TIMEOUTED, CANCELED, FINISHED.

Pass_Dev
Device type of the passenger, either AND or IOS.

Driv_Dev
Device type of the driver, there was found only AND or empty record.

The described raw data were later analyzed in the appendix A, processed and used for additional feature extraction which is described in the following section 4.3.

From the processing of the dataset we should especially mention the removing of the data records that was not making any sense. To this category belongs all the data records where was the "PassengerPositionAtOrdering" further than within heuristically set 5km from the "RequestPickupLocation". We can of course imagine situations when booking taxi with this kind of setting might be justifiable, but most of the time records like this seemed to be only users from places where the service is not available playing with the mobile application. Another data records that were removed were all the records where the "RequestPickupLocation", the "RideExpectedDestination" or the "DriverPositionAtInquiry" where outside of the Prague bounding box. The Prague bounding box was set as $49.9 < \text{latitude} < 50.3$ and $14.15 < \text{longitude} < 14.9$.

3.4.2 Driver activity data

Our second dataset that was available for this work was the driver activity dataset that contains recorded information about which driver was available in the certain time of the day. This dataset is especially useful for the later evaluation of the entire matchmaking mechanism.

The driver activity dataset is quiet simple and contains only few fields in comparison with the transaction dataset on the other hand it contains huge amount of records. Driver activity dataset contains following fields:

Driv_ID
ID of the driver.

Time_Stmp
Timestamp of the time when the record was recorded.

Driv_Pos_Lat
Latitude of the drivers position.

Driv_Pos_Lon
Longitude of the drivers position.

Driv_St
Current state of the driver, either AVAILABLE or SERVING.

As you can see the driver activity data records are very simple. Unfortunately the data had to be further preprocessed and merged with the transaction dataset. First of all we have created the dictionary where keys were particular minutes during the whole time period and values were lists of tuples containing the driver and it's position for all the drivers that were available in the particular minute. Using this dictionary we have created another dataset corresponding to the transaction data, but this time without label fields Driv_Act and Compl_State. The dataset we have created by iterating over all the requests and adding requests to all the drivers that were available in the same minute as in which the original request was sent.

When we have look at our system once again we can see that we can optimize the already existing mechanism by only focusing on it's sub-part that optimizes the request recipient selection in a way that the requests are being sent only to the relevant drivers. Very natural way of how to try to optimize the currently working system when we have available data is by the implementation of the machine learning models. Particularly in our case we have introduced one model that is called the driver response model and other model that is called the passenger's model. When we have more detailed look at the models we can realize that basically both models are more or less the same with the only difference that the drivers model should be learn on the Driv_Act labels and the passengers model only on the subset of the data where was already performed drivers accept action and as a label there should be used the field Comp_State. It might also be advantageous to keep both models separate for the case when we would potentially have some features that are related only to one of the models. We can especially imagine some additional features that might be added later on for the passenger's model. Particularly in our work we have especially focused on the driver response model, because here both we and the company that has provided us the data for this work see the biggest potential to improve the currently working taxi allocation system.

We have more in detail focused on all the machine learning techniques that we have experimented with in the following section 4. In the following section 4 we have also described all the methods that we have used for the feature extraction from the data and about how we have particularly learned and evaluated particular driver response models.

Chapter 4

Driver response model

In this section we have introduced the way how we have decided to improve the solution of the request recipient solution mechanism in comparison with currently deployed mechanism that is not using advantages of the machine learning, any other artificial intelligence methods nor any other smarter approach. First of all we have tried to formalize and describe the entire learning problem, then we have described all the mechanisms that we use in order to extract desired features from the dataset that was available to us. Features description has naturally been followed by the detailed description of the machine learning methods that we have experimented with in our work. At the end we have also explained the way how can be the learned model evaluated.

4.1 Learning problem description

For the optimization of the request recipient selection problem we have decided to introduce so called "driver response model". In the driver response model we have tried to particularly predict the probability of that there will be selected accept action for the passengers request from the data. In other words we have tried to predict $p(y_d|x_d, D_d, M_d)$ where categorical variables $y_{d_i} \in \{A_d, C_d\}$ and $x_{d_i} \in \{A_d, C_d\}$ where A_d stands for driver's ACCEPT action, C_d for drivers DECLINED action or for driver letting the passenger's request to be time outed (TIMEOUTED action). D_d represents the dataset and M_d represents our particular driver response probabilistic model.

Goal of this model is to determine the probability of driver to accept the given request. Based on this model's predictions we will be able to determine more precisely which drivers are potentially the best drivers with the highest probability of accepting the passengers request. Then we can basically only sort the drivers by the predicted probability and send the request to the N best drivers. In a consequence this optimized solution using the probabilistic classifier model should help us to generally decrease number of "spam" request sent to the driver and also to increase the probability that the passenger's request will actually be served by one of the drivers by actually finding some more relevant drivers than those that were selected by the currently working mechanism.

The particular types of models that predicts us the class with it's associated probability are called probabilistic classifiers. The reason why usage of this kind of classifier models instead of the standard classifier is advantageous for us is the that there is an industry driven need to better target several drivers with lower probability that is still the highest among all the drivers than to not send the request to any of the drivers. In other words it's better to spam less likely drivers than give up directly after receiving the passengers request seeing that we don't have fully relevant taxi driver for him at a current moment.

4.1.1 Probabilistic classification

At the beginning of this section we have identified that natural way of treating the problem when we need to predict the probability is using probabilistic classifiers. There exist various methods of probabilistic classification that are more in detail described in [35] from these methods the most natural and well known seems to be Naive Bayes which detailed description can be found in [36]. Unfortunately our problem does not fulfill the Naive Bayes condition that all the features should be independent. For example we can clearly see that for example the distance between taxi driver and passenger departure point depends on the taxi driver's and passenger's departure point.

As a second representative of the probabilistic classifier can be considered any ensemble method. Here the approach how to compute the final posterior probability is simple, we only need to divide the number of classifiers voting for particular class by number of total classifiers. As an example that should perform well for our data we can name random forest classifier that is more in detail described in [37]. Here we just have to be careful about the classifier training mechanism.

Third example of probabilistic classifier we can under certain conditions see in artificial neural networks (ANNs). Conditions that the ANN have to meet are more in detail described in [38], especially in section B.

Last, but not least method that we will consider for our work are support vector machines (SVMs) which is not probabilistic classifier in its nature, but there exist some methods of turning it into one. Very detailed description of these methods can be seen in [39].

The very last method that we have considered for our work is simple modification of the KNN, where we can simply compute the probability by division of the elements of certain class in the K nearest neighbors by the K. More details are described in [40].

4.1.2 Learning top K

Another very interesting problem for our work is so called learning top K problem. This is important for us mainly because in the request recipient part of our mechanism we have to select top K most relevant taxis and we can gain some inspiration from various learning top K approaches.

Majority of the work has previously been done in the field of document retrieval, which

is not a big problem for us, because documents are same as taxi requests usually represented as vectors. Some basic overview, where learning top-k is descried as a part of the learning to rank algorithm can be seen on the wikipedia[1]. Some work describing top-k as a part of the learning to rank algorithm was described in [41]. Learning top-k was investigated with it's evaluation framework in [42].

4.2 Evaluation framework of the driver response model

Evaluation of the probabilistic classifier like models can be done in a very simple way. First of all, because of we have only the classification to two classes we can set probability 0.5 as a threshold for classifying the records to particular classes. We have used this assumption and we have computed and recorded how many items from the training set with label ACCEPTED were classified as ACCEPTED and how many of them were classified as DECLINED. Then we have recorded the same for items with DECLINED label in the real world data.

Second important metric that we use for measuring the trained classifier performance is the brier score[2]. The biggest advantage of the brier score is that it also reflects very well how far was on average the predicted probability from the actual label.

The resulting measured values we have recorded in the following table.

	brier score	correct accepts	correct declines	accepts as declines	declines as accepts
no artificial features					
artificial features					

4.3 Features construction from the data

This section we have basically divided into the two parts, in the first part we have described the features that we have available directly from the data and in the second part those features that are from the extraction point of view little bit more interesting, features that we have derived or computed by our own from the data. During the selection of the features we have used our previously performed data visualizations that are available in the appendix A.

4.3.1 Original features from the data

For the training of the driver response model we can use directly several available data records, particularly we have used:

[1] <http://en.wikipedia.org/wiki/Learning_to_rank>

[2] <https://www2.physics.ox.ac.uk/sites/default/files/2011-06-09/2010_qjrms_136_1364_preprint_pdf_14112.pdf>

driver's ID
ID of the driver that was broadcasted by the passengers request, copied field Broad_Driv_ID.

passenger's ID
ID of the passenger, copied field Pass_ID.

timestamp
Timestamp of the time when the transaction was broadcasted, copied field Broad_At.

Then we have done first only small data preprocessing tasks that were usually using only one simple API call to obtain the desired value. Particularly we have used Python build in API for work with timestamps and remote ATG's API for computing distances and traveling time between two geographical points. At the end we have created following features:

hour
Feature that represents the hour of the day and that was computed from the Broad_At timestamp.

day
Feature that represents the day of the week that was again computed from the Broad_At timestamp.

journey length
Distance of the journey with passenger that was computed from the fields Req_Pick_Loc and Ride_Exp_Dest.

journey time
Time of the journey with passenger which was again computed from Req_Pick_Loc and Ride_Exp_Dest.

journey to pick up length
Feature that represents the distance that the driver has to travel to pick up the passenger it was naturally computed from the fields Driv_Pos_At_Inq and Req_Pick_Loc.

journey to pick up time
Time needed to travel the journey to pick up the passenger it was again computed from the fields Driv_Pos_At_Inq and Req_Pick_Loc.

4.3.2 Computed or derived features

To even improve our models we have decided to introduce some other relevant features that are potential hidden in the data and there is needed little bit more complicated computations on the data to obtain these features.

For the first set of features we have heuristically set the geographical point with latitude 50.0880400 and longitude 14.4207600 that is representing the center of Prague. Then we have computed straight line distances and azimuths from this point to the various important points for us. We have obtained following features:

drivers position distance
Represents the straight line distance between the Prague center point and the drivers position, it uses the Driv_Pos_At_Inq field.

drivers position azimuth
Represents the azimuth from the city center to the drivers actual position obtained again from the field Driv_Pos_At_Inq.

passengers position distance
Feature that stands for the straight line distance between the Prague center point and the passengers desired pick up position, it uses the Req_Pick_Loc field.

passengers position azimuth
Feature that describes the azimuth from the city center to the passengers desired pick up position obtained again from the field Req_Pick_Loc.

passengers destination distance
Represents again the distance from the distance from the heuristically set center of Prague, but this time to the passengers desired destination point. It uses Ride_Exp_Dest field.

passengers destination azimuth
Feature that describes the azimuth from the city center to the passengers desired destination, it uses the field Ride_Exp_Dest.

Our next set of artificiality constructed features that we have introduced were ids of the geographical points cluster. For computation of this we have used K-means unsupervised learning method. We have run this separately for drivers positions, passengers start positions and for passengers destination positions. As a distance measure for K-means we have used the Euclid distance. The only problem of this approach was how to select the appropriate number of clusters. Here we have decided for trying experimentally K=10, 20 and 50. Particularly we have computed following features:

ID cluster10 driver position
This represents the ID of cluster to which there belongs the drivers position when the number of clusters for K-means algorithm was set to 10. There was used the field Driv_Pos_At_Inq.

ID cluster20 driver position
This feature analogically represents again the ID of the cluster where the drivers position belongs, but this time for K set for 20. Again there was used the field Driv_Pos_At_Inq.

ID cluster50 driver position
Again even this feature corresponds again to the ID of the cluster where the drivers position belongs, but this time for K set for 50. Again even in this case there was used the field Driv_Pos_At_Inq.

ID cluster10 passenger pick up
This represents the ID of cluster to which there belongs the passengers requested pick up point when the number of clusters for K-means algorithm was set to 10. There was used the field Req_Pick_Loc.

ID cluster20 passenger pick up
This feature again represents again the ID of the cluster where the passengers requested pick up location belongs, but this time for K set for 20. Again there was used the field Req_Pick_Loc.

ID cluster50 passenger pick up
Again even this feature corresponds again to the ID of the cluster where the passengers desired pick up location belongs, but this time for K set for 50. Again even in this case there was used the field Req_Pick_Loc.

ID cluster10 passenger destination
This feature represents the ID of cluster to which there belongs the passengers desired destination point when the number of clusters for K-means algorithm was set to 10. There was used the field Ride_Exp_Dest.

ID cluster20 passenger destination
This feature analogically represents again the ID of the cluster where the passengers desired destination geographical point belongs, but this time for K set for 20. Again there was used the field Ride_Exp_Dest.

ID cluster50 passenger destination
Again even this feature corresponds again to the ID of the cluster where the passengers desired destination location belongs, but this time for K set for 50. Again even in this case there was used the field Ride_Exp_Dest.

Another artificially created set of features for which we have used K-means method were segmentation of the drivers. In this case the data were needed to be preprocessed for the run of the K-means. First of all we had to create new data set in which new record looked as follows $d_i = (h_{j_d}, d_{j_d}, pos_{hdj_d}, a_d, a_{hdj_d})$ where h_{j_d} represents average number of requests sent to this driver in a certain hour of the day, d_{j_d} represents average number of requests sent to the driver in a certain day of the week, pos_{hdj_d} position of the driver based on the time of the day and based on the day in a week, a_d represents the total drivers accept ratio and a_{hdj_d} which represents the drivers accept ratio based on the time of the day and day in a week. Again same as in the previous case the problem was, how to set properly the number of clusters, in this case we have experimentally decided for K=5, 10 and 20. From this we have obtained following features:

driver's cluster5 ID
This feature represents the ID of cluster to which there belongs the driver when the number of clusters for K-means algorithm was set to 5. There were used various fields of the original dataset for the K-means clustering.

driver's cluster10 ID
This feature analogically again represents the ID of the cluster where the particular driver belongs, but this time for K set for 10. Again there were used various fields of the dataset.

driver's cluster20 ID
Again even this feature corresponds again to the ID of the cluster where the particular driver belongs, but this time for K set for 20. Even in this case there were used various fields of the original dataset.

The last new introduced features on the side of the driver was computed number of total requests and the number of accepted features by the driver within the last hour and within the last day.

last hour accepts
Feature that represents the total number of accepted requests from the driver during the last hour.

last hour requests
Represents the feature that stands for the total number of requests sent to the particular driver during the last hour.

last day accepts
Feature that represents the total number of accepted requests by the driver during the last 24 hours.

last day requests
Represents the feature that say how many requests were sent to the particular driver during last 24 hours.

4.3.3 Final feature vector

For better clarity we have collected all the mentioned features from the two sections above and recorded them to the following table 4.1.

4.3.4 Other considered and not used features

There were also several features that we have experimented with, but which we at the end have not used. The reason why these features were not used was that all of these features seemed to be reasonable, but it was currently impossible to clearly extract them from the data. On the other hand we still strongly believe that in the future together with the growing market penetration by the application it will be possible to extract and use below mentioned features. So we are proposing them as something like possible future work.

position on the taxi stand
Our idea was to use the position on the taxi stand as a feature, but unfortunately there were clearly seen taxi stand spots in the data very rarely, because the application covers only certain percentage of all the available taxi drivers in Prague.

feature name	**type**
driver's ID	categorical
passenger's ID	categorical
timestamp	numerical
hour	numerical
day	numerical
journey length	continuous
journey time	continuous
journey to pick up length	continuous
journey to pick up time	continuous
drivers position distance	continuous
drivers position azimuth	continuous
passengers position distance	continuous
passengers position azimuth	continuous
passengers destination distance	continuous
passengers destination azimuth	continuous
ID cluster10 driver position	categorical
ID cluster20 driver position	categorical
ID cluster50 driver position	categorical
ID cluster10 passenger pick up	categorical
ID cluster20 passenger pick up	categorical
ID cluster50 passenger pick up	categorical
ID cluster10 passenger destination	categorical
ID cluster20 passenger destination	categorical
ID cluster50 passenger destination	categorical
driver's cluster5 ID	categorical
driver's cluster10 ID	categorical
driver's cluster20 ID	categorical
last hour accepts	numerical
last hour requests	numerical
last day accepts	numerical
last day requests	numerical

Table 4.1: Table of used features

driver is on taxi stand or not
Our second idea was to at least determine if the driver is on the stand spot. But unfortunately determining where is the taxi parking just from the transaction data was not possible. Here would probably help to use some additional dataset containing locations of all the taxi stand positions.

standing or cruising state of the driver
Again our idea was to determine if the driver was standing or cruising. The problem with this feature was that it was very hard to determine if the driver is moving or not from the drivers activity data since it seemed that the times periods when drivers were recording their positions varied a lot.

4.4 Used machine learning methods

We have experimented with several machine learning models learned on the either the data set with all the features or on the data set with just the features that were introduced in the section original features from the data.

4.4.1 Naive Bayes

As the first and very natural approach of how to build the probabilistic classifier we have selected Naive Bayes method as it is described in [43]. Particularly we have first trained two "submodels", the Multinomial naive bayes model from the categorical features and Gaussian naive bayes model from the continuous features. Then we have learned final Gaussian naive bayes model on the probabilities predicted by both of the models as it is recommended in [3].

The biggest disadvantage of using the naive bayes classifier is that it assumes that all the features are independent, which we can clearly see is not true in case of our data set. For example we can say that usually the distance and traveling time between two points are highly correlated features.

4.4.2 K nearest neighbors

Another very easy to interpret classifier that can be turned to the probabilistic classifier is K nearest neighbors method. Which was again described in [43]. In this case we have used the Euclid distance as a measure of the distance between points. Then we have used the standard cross validation to obtain optimal parameter K. We were able to see that this K was usually estimated to numbers from 20 to 40. This number varied based on the features that were covered by the data set that we have used for learning. Then we have determined the probability of acceptance or reject by simply computing the number of records classified as accept in the K best records divided by the total number K.

[3] <http://stackoverflow.com/questions/14254203>

4.4.3 Decision tree forest

The most sophisticated probabilistic classifier method that we have used were decision tree forests. In this case there was used the implementation of the random forest classifier from the scikit learn library[4]. Which performs combining the particular tree classifiers by averaging their probabilistic prediction, instead of letting one tree vote for one particular class.

On the other hand the disadvantage of the random forest classifier from the scikit learn library is that it does not perform pruning algorithms during the learning of the particular trees. This fact tends to cause overfitting by particular trees used for the classification. One possible way, how to prevent particular trees from the overfitting is to manually set the parameters max_depth that determines the maximal allowed depth of the tree, $min\_samples\_split$ which determines the minimum number of samples that are required to split an internal tree node and $min\_samples\_leaf$ that determines what is the minimum number of samples per newly created leafs.

Naturally the last parameter that we had to determine was number of decision trees that we have used for learning. Again same as in the case o K-nn we have used cross validation to set the optimal parameters for the data sets from the different periods of the time.

4.5 Conclusion on the proposed driver response models

In this section we have described various feature extraction tasks that we have performed. More over we also have identified several possible features by which we believe that we will be able to even improve our models in the near future when there will be higher market penetration by the taxi booking application.

Then we have also investigated and selected methods of how to learn the model that will predict certain class with it's associated probability. At the end we have also identified Naive Bayes, K nearest neighbors and decision tree forest as three methods which we will use as a driver response models. The main reason of choosing particularly these methods was that Naive Bayes and K nearest neighbors are very easy to interpret as a probabilistic classifier model and decision tree forest is based on the related literature is supposed to perform very well in terms of the prediction quality. In the next sections we will show more details about how we have implemented the learning of the mentioned models and also how has the particular models performed in the evaluation section.

[4] <http://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees>

Chapter 5

Implementation

As it was mentioned in the previous sections the whole system was implemented in Python as several independent scripts that can be run separately. Our system also advantages of the Python built in libraries such as NumPy or Scikit learn [44] which provide us very powerful and easy to use tools for the data analysis and machine learning models training.

5.1 Python

We have selected Python because it is widely used general purpose programing language for which there was already implemented a huge number of various open source libraries for machine learning and artificial intelligence. Our scripts were developed particularly in the Python version 2.7. which assures us better compatibility with the most of the frequently used machine learning libraries.

Another reason for using Python was it's very good readability a simplicity to write the code together with possibility to implement the critical operation in C/C++ which we can notice in mentioned libraries Numpy and Scikit learn. From this you can also notice that critical for us is to use CPython, the default and most widely used implementation of Python.

5.1.1 NumPy

As a first of the libraries that were used by our scripts we should mention NumPy, which is the extension to Python that enables to perform very similar operations as Matlab programming language in Python. In other words it enables easy to use optimized computation with matrices and vectors.

5.1.2 Scikit learn

Another very important Python library for us is Scikit learn. This library provides easy to use, very robust and stable stack of various machine learning algorithms. These algorithms are also optimized in terms of the speed performance and its critical parts are usually implemented in C/C++. Particularly for our work we have used below described algorithms.

5.1.2.1 Naive Bayes

The first machine learning algorithm that we have used from Scikit learn was Naive Bayes classifier [1]. It is better to say that is more than just one algorithm the whole family of algorithms. We have particularly used Gausian and the multinomial Naive Bayes.

5.1.2.2 K nearest neighbors

Another Scikit learn algorithm that was used in our scripts was K nearest neighbors [2]. We have particularly used the standard K nearest neighbors. Advantage of the Scikit learn implementation was that it has let us quiet easily to try out both approaches, when the weights for one particular class were distance and uniform value for each point.

5.1.2.3 Decision tree forests

Probably the most complex machine learning algorithm from Scikit learn that we have used was decision tree forest [3]. Particularly we have used the algorithm called RandomForest-Classifier which trains a certain number of decision trees classifiers on the various sub-sets of the input data and sub-set of input data features. There is used averaging of the trees probabilistic prediction instead of the single vote assigned for each of the trees.

The biggest disadvantage of the Scikit learn RandomForestClassifier implementation is that there is not implemented any algorithm that supports particular decision tree classifier pruning. For this reason there had to be used the Grid search with cross validation to determine the particular parameters to prevent the classifier from overfitting.

5.1.2.4 K-means clustering

The last important algorithm from Scikit learn that was used in our scripts is K-means clustering [4]. This algorithm performs simple data clustering in two steps. At the beginning there are randomly selected K points from the data as means. Then in the first step there are computed closest means for the each data point. Then in the second step there are recomputed new means. As a criterion that should be minimized we use standard Euclidean distance.

As you could have noticed from the description of the K-means algorithm, it is an algorithm with no guaranties of returning the optimal solution. Thus in the Scikit learn implementation of the K-means there is a possibility to chose number of how many times is K-means with different initial points as cluster centroids is run.

[1] <http://scikit-learn.org/stable/modules/naive_bayes.html>
[2] <http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification>
[3] <http://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees>
[4] <http://scikit-learn.org/stable/modules/clustering.html#k-means>

5.1.2.5 Grid search

Last functionality from Scikit learn that was widely used in our work was grid search [5]. Grid search is basically the implemented combination of searching for the optimal parameters from "grid" of all the possible parameters with the cross validation. We have used this method to estimate the optimal parameters for our supervised classifiers.

5.2 Particular scripts description

As we have mentioned in the previous section our program consists of several scripts that can be run independently. In this section we will describe all the important scripts one by one. In the figure 5.1 you can see, how it is recommended to run the scripts to train and evaluate models on the provided data. There is no single pipeline script, because of very long time needed top run some of the scripts, particularly computing distances and traveling time via ATG's API or grid search for particular model parameters.

5.2.1 Data preprocessor

In the data preprocesor there are all the possible functions that can select various data subsets, for example data subset for certain amount of time, data subset for all the entries within the Prague bounding box. This script also contains functions that are computing distances and the traveling time for all the desired geographical points pairs.

5.2.2 Feature extractor

In this script there are functions that extracts only the important features from the data matrix that contains also various other data fields that are not further used for the classifier learning. It also performs date conversion to the weekday and hour format and also geographical point to the polar coordinates.

5.2.3 Feature builder

In the Feature builder we can find creation of all the features that are created by the unsupervised learning methods, such as K-means clustering of the passenger's origin and destination coordinates, driver's position coordinates or K-means clustering of the particular Taxi drivers and passengers.

5.2.4 Drivers model

In this script we have all the classification models learning related functions. We have implemented all of the classifiers as separate classes that we are able to persist in a file for further use. We have also implemented the possibility to compute the Brier score for each of the probabilistic classification models.

[5] <http://scikit-learn.org/stable/modules/grid_search.html>

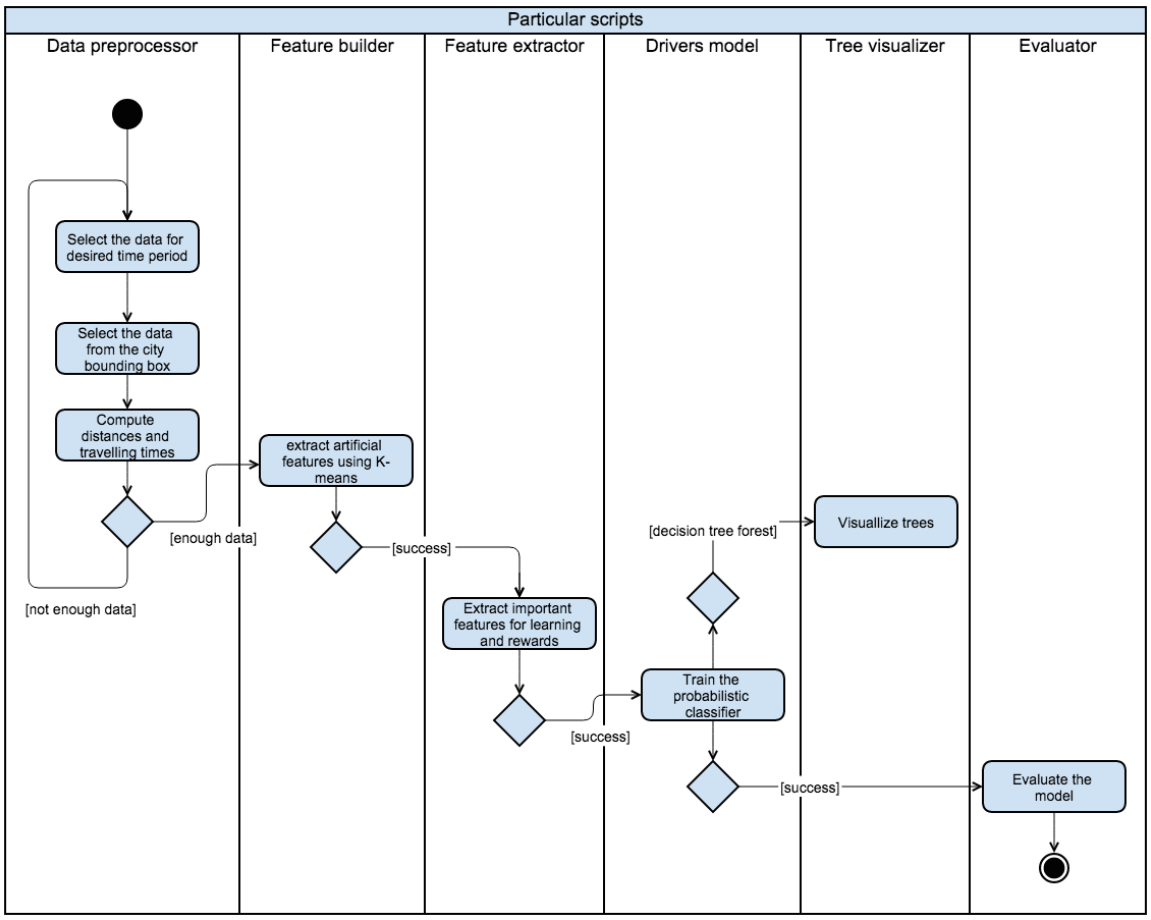


Figure 5.1: recommended script run order

5.2.5 Tree visualizer

Tree visualizer is only a small script that enable us to convert each of the decision trees to the .pdf format. This can be useful for visual control the learned decision trees.

5.2.6 Evaluator

Last and very important script is the evaluator which contains various functions to perform evaluation of the learned models. Evaluation functions that can be found in this script were described in the section evaluation framework and measured results will be described in the following section evaluation.

Chapter 6

Evaluation

The evaluation of our models or more over evaluation of the whole matchmaking mechanism is quiet complicated and even more it seems to be even impossible to perform a perfect evaluation based only on the historical data. So we can not clearly say that our evaluation would clearly tell us that our proposed model works actually better than the already working system selecting the taxi drivers based on some heuristics. For very precise evaluation of our system there would be needed to perform at least some kind of additional A/B testing involving the run of our model in the real system.

First of all we describe, how we perform the evaluation, then we describe the metrics that we have measured. After this we also describe the way, how we have set the parameters of our models that were used in our mechanism. In the next section we measure, record and visualize the important measured values and as a last part of this chapter we introduce few information about how should be performed the following A/B testing and what should be measured during this kind of testing.

6.1 Evaluation scenario

Since we had available the consecutive transactional data for August and September 2014 we have have decided to split the data for training and testing data in a way to have one testing week or in other words the last 7 days of September 2014 we have used as a test dataset and the rest of the data we have used as a training data.

6.1.1 Driver response model evaluation scenario

First part of our evaluation has used only the transactional data described in the section 3.4.1. As it was said before we have split the data for the training and the test dataset. Then we have found optimal parameters using previously described grid search from the scikitlearn. After that we have again learned the model using whole training dataset and measured the evaluation metric on the testing data. We have performed all these evaluation steps twice, first of all for the models learned on the data without new artificially computed features and once on the dataset with all the derived features from the dataset that we have described in the section 4.3.2.

6.1.2 Mechanism evaluation scenario

Second part of the evaluation deals with the evaluation of the integrated model within the mechanism. As a data input we have used the newly created dataset containing merged both the transactional data with the driver activity data as it was described in the section 3.4.2. Here we must say that there were some obvious inconsistencies in the data. Thus we have decided to remove all the records that were clearly inconsistent. As an example we can mention case when there was certain driver available based on the transactional data, but not available based on the drivers activity data or when in the transactional data has appeared the driver that whose ID was never spotted in the driver activity data. At the end we end up with only 4245 valid data records with label accept and 22614 data records with label decline, which is in the both cases slightly below 60% of the original records in the transaction data.

Again as in the model evaluation we have also for the mechanism evaluation we have performed all of the evaluations for all the classifiers, once without artificially created features and once with artificially created features that were described in the section 4.3.2.

In the evaluation method that we have introduced we are basically recording only the number of records in the table which should at least show that the learned model is not completely wrong. For better visualization we have then also computed percentage values from all the records and the percentage of accepts and declines from the real world transaction data that were selected.

6.2 Evaluation metrics

Again in this section we have first introduced the evaluation metrics for the driver response model evaluation and the for the evaluation of whole matchmaking mechanism using the integrated driver response model.

6.2.1 Evaluation metrics for the driver response model

In this section we were particularly measuring five different metrics. First of all we were computing the brier's score of the learned model on the tested data. Then we have computed four values where first was number of accepts classified as accept with probability higher than 0.5, then number of accepts classified to be accept with probability smaller than 0.5. The same we have measured and recorded for declines. More details about metrics can be seen in the section 4.2.

6.2.2 Evaluation metrics for the matchmaking mechanism

Measured and recorded metrics for the matchmaking mechanism corresponds pretty much with the proposed evaluation framework in the section 3.3.2. In other words we were measuring all the tabular data of how many times there was predicted to one of the four categories

	number of trees	max depth	min samples split	min samples leaf
without artificial features	125	30	4	2
with artificial features	195	45	1	2

Table 6.1: Decision tree forest parameters table

for all the data that had label accept, decline or data where we don't know the label. Our four categories to which the particular drivers could have been predicted were accept and selected (accept), decline and selected (decline), accept and not selected (AnS) or decline and not selected (DnS). For better clarity of the measured result we have also recorded the percentage from all the evaluated requests in the parenthesis.

6.3 Model configuration

As it was described in the previous sections, we have used so called grid search from the scikit learn library that internally uses the cross validation to find the best parameters for our K nearest neighbors and decision tree forest models.

6.3.1 Estimated parameters for K nearest neighbors

For the K nearest neighbors was the situation quiet easy. The only parameter that we had to determine was K, the number of neighbors that should be taken into the account.

We have performed the grid search first for the model learned on the data without artificially created features. Where the optimal K was selected as 21. And then we have performed the grid search again, but this time for the data with artificially created features, where the optimal K found as 30.

6.3.2 Estimated parameters for decision tree forests

For decision tree forests become the situation slightly more complicated, because of several other parameters than just number of trees that had to be estimated. First of all it was the number of trees in the forest, then maximal depth of the tree, minimum samples needed for the split, and minimum samples needed in the leaf.

We have first performed the grid search for the model learned on the data without artificially created features. The optimal parameters are shown in the table below. Then we have again performed the grid search for data with artificially created features and again the optimal parameters can be found in the following table 6.1.

6.4 Measured results

In this section we have separately performed the evaluation for the driver response model using various machine learning techniques. Then we have separately evaluated the whole

	brier score	correct accepts	correct declines	accepts as declines	declines as accepts
no artificial features	0.2594	2586	36052	4597	3765
artificial features	0.2834	2875	34931	4308	4886

Table 6.2: Evaluation of the Naive Bayes classifier

	brier score	correct accepts	correct declines	accepts as declines	declines as accepts
no artificial features	0.3061	1022	36890	6161	2927
artificial features	0.3045	1404	35852	5779	3965

Table 6.3: Evaluation of the K nearest neighbors

mechanism that was relaying on the model. We have also summarized conclusion for every part of the evaluation.

6.4.1 Driver response model evaluation

First and the easiest way how we possibly could evaluate probabilistic classifier model was to divide the data set to the train and test data and then compute the Brier score on the test data. Another very simple way of the evaluation is to compute the number of correctly predicted labels. Basically we are just comparing the labels of the taxi drivers for that we have the label available with our predicted labels. Unfortunately we have recorded label only for the fraction from the total amount of drivers in the system.

6.4.1.1 Results for Naive Bayes classifier

As we have said before first we have performed the evaluation of the model learned on the data without artificially crated features and then the same evaluation of the model this time learned on the data containing also the artificially created features. All the measured values were recorded in the following table 6.2.

6.4.1.2 Results for K nearest neighbors

Analogously as for the Naive Bayes even for K nearest neighbors we have learned one model on the data without artificially created features and one on the data containing this set of features. Again we have recorded the measured values to the following table 6.3.

6.4.1.3 Results for decision tree forest

Same as for the previous two probabilistic classification models we have again even for the decision tree forests learned two models, one for data without artificially created features and one for data with this set of features. Again the measured results were recorded in the following table 6.4.

	brier score	correct accepts	correct declines	accepts as declines	declines as accepts
no artificial features	0.2597	3087	35254	4096	4563
artificial features	0.2489	3414	34320	3769	5497

Table 6.4: Evaluation of the decision tree forest

6.4.1.4 Conclusion on the prediction quality measurements

First of all we have performed the visualization of the measured values in the figure 6.1. From the probabilistic classifier accuracy measurements and it's visualization we can see that the best performing model for our data was the decision tree forest learned on the data containing also the artificially constructed features. On the other hand we can see that the Naive Bayes model was doing also surprisingly well and in the case when we did not used artificially created features it has performed more or less the same as the decision tree forest. Unfortunately by introducing new features that were probably not independent on others has quality of the Naive Bayes remain more or less the same and we can probably not expect any significant improvement of the Naive Bayes model by introducing new features. The worst performing model in terms of plain model evaluation was definitely K nearest neighbors.

6.4.2 Matchmaking mechanism evaluation

In this section we have focused on the evaluation of the whole proposed mechanism using our model. Our goal is to select N best taxi drivers based on the model and then evaluate this selection. From this measurements it should be possible to see how much are selections using our model different from already existing mechanism based on some heuristics and it was also possible to see that our model actually works, but unfortunately it was very difficult to evaluate if our model actually works better than the current mechanism.

We have performed the evaluation using tables introduced in the evaluation framework section 3.3.2. We have filed this tables with total occurrences for the whole period of the last 7 days of September 2014. In parenthesis we have recorded the the percentage from the total amount of accept and reject records that was available also in the driver's activity data. We have also learned all of the models twice. Firstly using the data without artificially created features and then on the data containing both direct features from the data and also the artificially constructed features as it is described in the section 3.4.2.

In addition to the evaluation on the total data using tables we have decided to compute and record the percentage of accepts from the real world data that were selected by our model ans the percentage of real world declines that were selected by our model, because these number are very interesting for us from the matchmaking mechanism point of view.

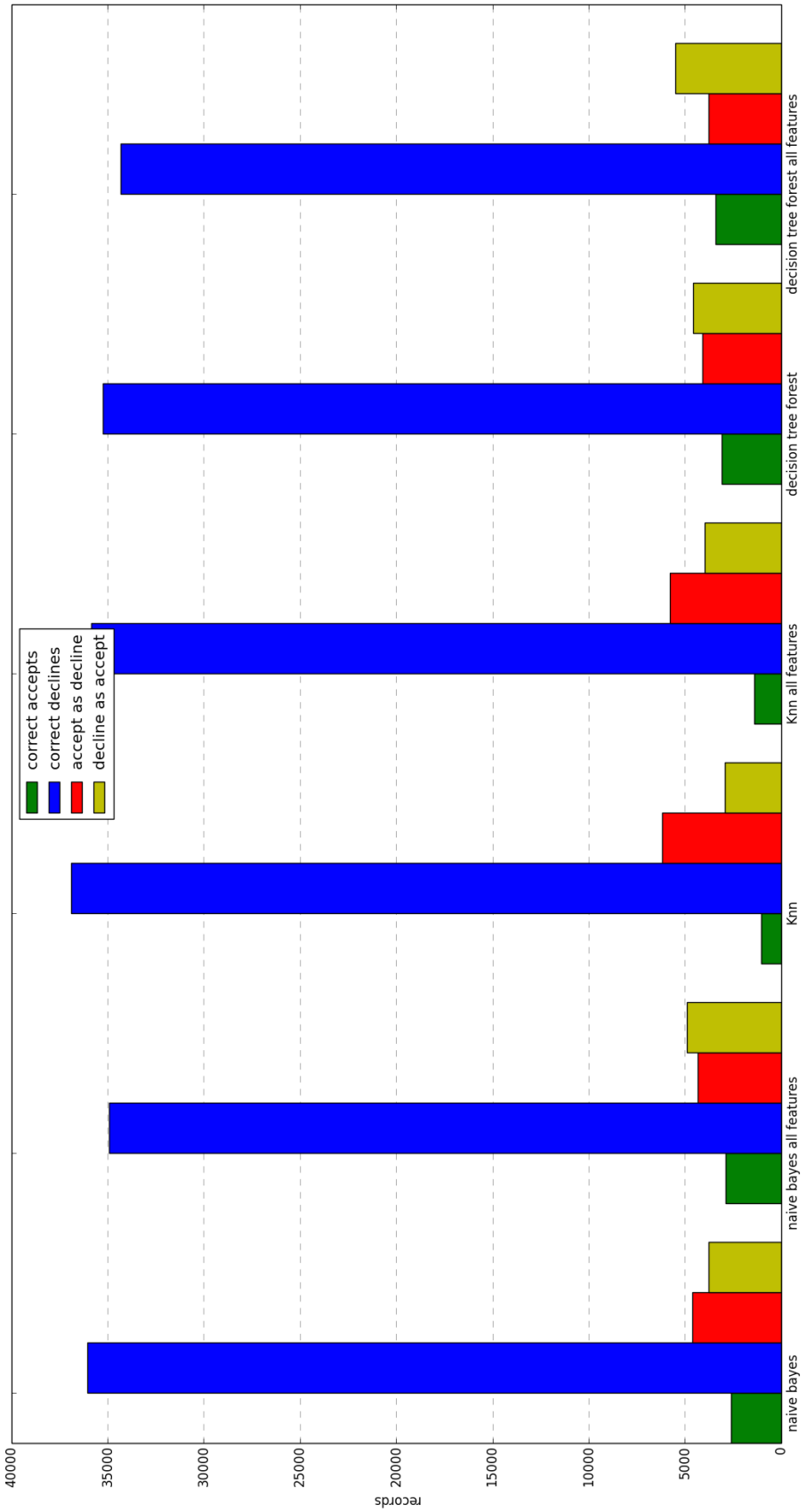


Figure 6.1: Number of classifications per model

		Real world		
		Reject	Accept	Don't know
	Reject	3486 (1.42%)	735 (0.30%)	36183 (14.77%)
Our model	Accept	327 (0.13%)	30 (0.01%)	6220 (2.54%)
	RnS	18436 (7.53%)	3443 (1.41%)	170638 (62.16%)
	AnS	336 (0.14%)	37 (0.02%)	5099 (9.38%)

Table 6.5: Table of mechanism evaluation using Naive Bayes without additional features

		Real world		
		Reject	Accept	Don't know
	Reject	3519 (1.44%)	739 (0.30%)	37548 (15.32%)
Our model	Accept	255 (0.10%)	19 (0.01%)	4901 (2.00%)
	RnS	18658 (7.62%)	3468 (1.42%)	172456 (70.39%)
	AnS	182 (0.07%)	19 (0.01%)	3235 (1.32%)

Table 6.6: Table of mechanism evaluation using Naive Bayes with additional features

6.4.2.1 Results for the Naive Bayes classifier:

First of all we have recorded measured model performance values for model learned on the data without artificially created features in the following table 6.5. You can see total number of records and percentage of total records in the parenthesis. Afterwards we have computed that on average there was selected 18.02% accepts from all the accepts that could have been potentially selected and 16.86% of declines from all the declines for that we had recorded decline in the transaction dataset.

Then we have performed exactly the same measurements, but this time for the model learned on the data with artificially created features. The measured values we have recorded in the table 6.6. Then we have again computed the average per request percentages with the result that there was selected 17.86% accepts from all the accepts that could have been potentially selected and 16.70% of declines from all the declines that could in the transaction dataset that could have been selected.

6.4.2.2 Results for K nearest neighbors:

Again as in case of the Naive Bayes even in here we have first computed and recorded the measurements for the model learned on the data without artificially created feature. The data can be seen in the following table 6.7. In this case our measured average accept selection percentage was 18.42% from all the accepts available in the real world data and average decline percentage was 18.10% again from all the declines available in the real world data.

Then we have again performed the same evaluation, but this time for the model trained on the data with artificially created features. Again the measured values were recorded in

		Real world		
		Reject	Accept	Don't know
	Reject	3838 (1.57%)	740 (0.30%)	40185 (16.40%)
Our model	Accept	248 (0.10%)	42 (0.02%)	1928 (0.79%)
	RnS	18436 (7.52%)	3426 (1.40%)	175129 (71.48%)
	AnS	92 (0.04%)	37 (0.02%)	898 (0.37%)

Table 6.7: Table of mechanism evaluation using Knn without additional features

		Real world		
		Reject	Accept	Don't know
	Reject	4261 (1.74%)	891 (0.36%)	40823 (16.66%)
Our model	Accept	166 (0.07%)	18 (0.01%)	822 (0.34%)
	RnS	18077 (7.38%)	3291 (1.34%)	175792 (71.75%)
	AnS	110 (0.04%)	45 (0.02%)	703 (0.29%)

Table 6.8: Table of mechanism evaluation using Knn with additional features

the following table 6.8. Afterwards we have again computed the average accept selection percentage as 21.41% from all the available data records with accept label and average decline selection percentage as 19.58% again from all the available real world data with the label decline.

6.4.2.3 Results for decision tree forest:

Same as in the previous two cases even for the decision tree forest classifier we have used the same evaluation methodology. Again first we have measured the values for the model trained on the data without artificially created features. This measured values can be seen in the following table 6.9. Afterwards we have again computed the average selection percentage. This time as 12.51% for accepts, again from all the accepts that were available in the transactions dataset and average decline percentage as 15.90% from all the data with label decline in the transactions real world data.

Then we have again performed the same evaluation, but this time for the model trained on the data with artificially created features. The measured values were recorded in the following table 6.10. Then we have again computed the average accept selection percentage as

		Real world		
		Reject	Accept	Don't know
	Reject	1080 (0.44%)	119 (0.05%)	10138 (4.14%)
Our model	Accept	2516 (1.03%)	414 (0.17%)	32714 (13.35%)
	RnS	17610 (7.19%)	3421 (1.40%)	152301 (62.16%)
	AnS	1408 (0.57%)	291 (0.12%)	22987 (9.38%)

Table 6.9: Table of mechanism evaluation using Decision tree forest without additional features

		Real world		
		Reject	Accept	Don't know
	Reject	1171 (0.48%)	242 (0.10%)	9148 (3.73%)
Our model	Accept	2174 (0.89%)	535 (0.22%)	33711 (13.76%)
	RnS	16764 (6.84%)	3097 (1.26%)	145598 (59.43%)
	AnS	2505 (1.02%)	371 (0.15%)	29683 (12.12%)

Table 6.10: Table of mechanism evaluation using Decision tree forest with additional features

18.30% from all the available accepts in the real world data and the average decline selection percentage as 14.79% from all the available real world data with the decline label.

6.4.2.4 Conclusion on the mechanism evaluations

When we have a look at the measured values in the mechanism performance measurements we can clearly see that number of accepts and declines does not match with the data from the model prediction quality measurements. This is caused by the usage of the combination of the transaction and the driver activity data. Unfortunately driver activity data were available only in the separate data set and we could have noticed that both datasets did not match as accurately as it would be needed for better evaluation of the system. Unfortunately this fact causes that the measurements in this section were probably more just a guideline and should be interpreted very carefully. Also because of this we even recommend before implementing this system into the production to repeat this experiment in a real world setting aside of the production code.

Nevertheless we have at least visualized the real world accept selection percentages and real world declines percentages in the figure 6.2. Then we have also visualized all the data from the tables in the figure 6.3, where we have focused only on the columns where we had available the label for the data which are particularly interesting for us.

Since we are interesting in high percentage of selected accepts and the low percentage of selected rejects/declines, we can again see that the best performing model seems to be again the decision tree forest trained on the data with artificially created features, where the difference between accepts and declines was the highest. But unfortunately this time the observation is not that obvious as it was in the evaluation of only the driver response model. We can again see that the Naive Bayes models were performing quiet well too, especially when we did not have available additional artificially created features from the data. We can also see that K nearest neighbors performed surprisingly well in selecting accepts, but unfortunately also quiet badly in selecting declines. On the other hand we must say that the improvement in comparison with the original taxi selection system is probably not as big as we would like to achieve. From the data it seems that usually when there were selected taxis with outcome decline, then there was probably not many other taxis that would score better and that could improve the final selection. This can be seen from the fact that usually there are high numbers in reject not selected and accept not selected for our best performing models. The other problem is that we have majority of the data in the column where we don't know the label and thus we don't know if the selection of such a record was actually improving of

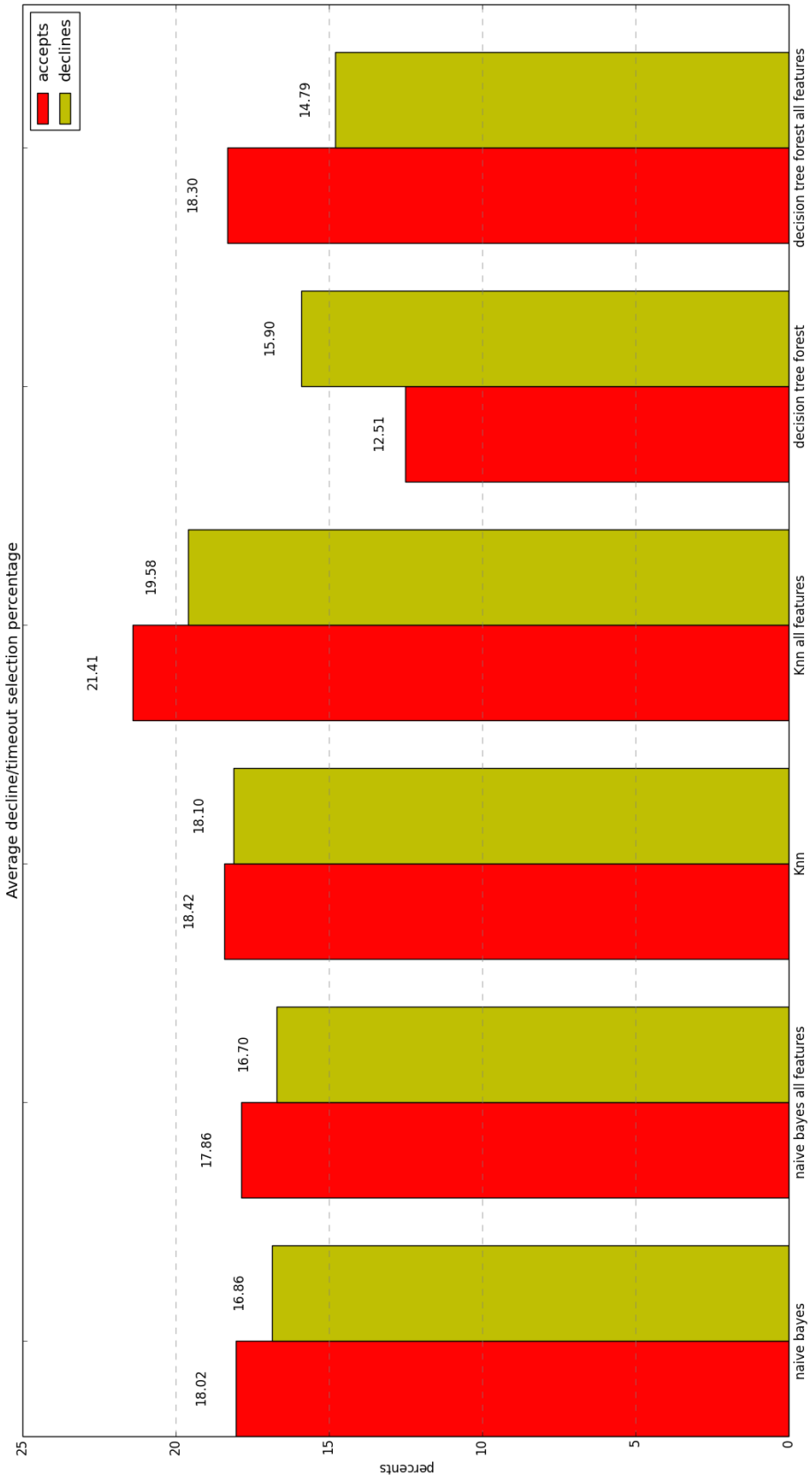


Figure 6.2: average accepts and declines percentage

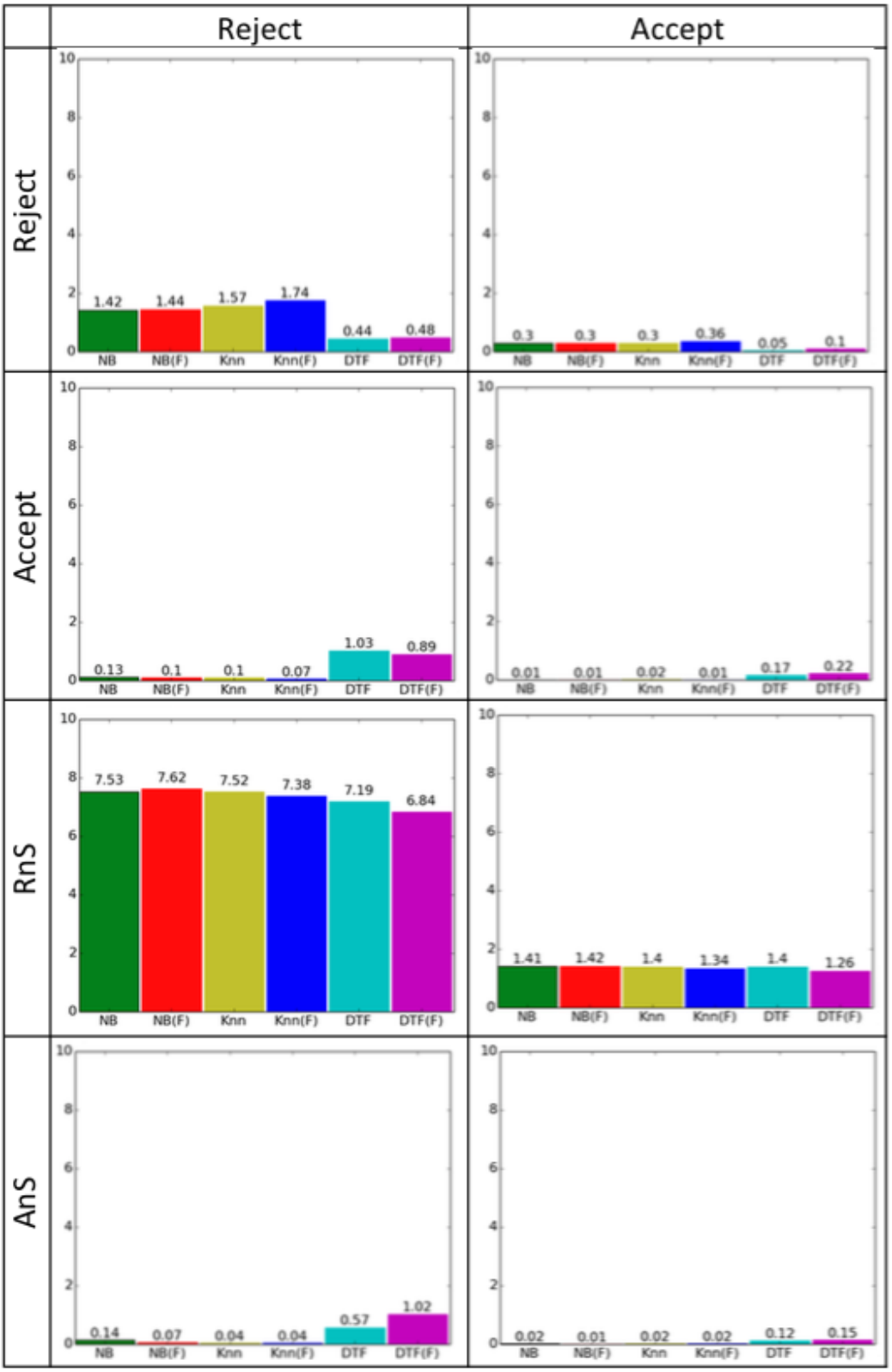


Figure 6.3: Visualized tabular data

the situation that we have recorded in the data or if we would have made the situation even worse.

6.5 Recommendations towards the A/B testing

In this section we have tried to describe what should be the following steps towards implementing our mechanism in to the production.

6.5.1 Re-run of the table evaluating method

First of all we strongly recommend to run the mechanism evaluating method using tables once again, but this time aside of the production code with exactly the same space of the taxi drivers. Here we should particularly focus on that if our mechanism select real world requests with the accept outcome and if our model drops real world requests with the decline outcome, in other words we should focus especially on the first two rows of the evaluation table.

6.5.2 Starting with our mechanism in the production

We recommend then to start with the implementation of our mechanism slowly and piecewise. First of all the mechanism should be used only for allocating the jobs for the taxi drivers in certain small percentage of cases and then it should be measured if the acceptance rate among drivers selected by the mechanism is significantly higher. Then the percentage of the request allocated by our mechanism might be slowly increased. Than it should be again measured and verified that the usage of our mechanism really brings the higher acceptance rate by the taxi drivers. The process of increasing the number of requests should be run several times based on the measured results. Disadvantage of the standard A/B testing like this might be quiet a long time that will be required to determine if our predicted model really improves the current matchmaking mechanism.

Another option for us is to use some from slightly more advanced methods for the hypothesis testing. Example of very nice approach might be so called Bayesian Bandit[1][2] which is based on comparing the beta distributions and that is very easy to implement.

[1] <https://www.chrisstucchio.com/blog/2013/bayesian_bandit.html>
[2] <http://www.mlguru.cz/bayesovsky-bandita-chytrejsi-a-levnejsi-ab-testovani/>

Chapter 7

Conclusion

In this work we have first visualized various interesting aspects of the data set that was provided to us. Than we have selected and constructed several new interesting features that helped us to create and further improve the mechanism, later we have introduced one possible system of how to design the passengers allocation mechanism based on the data driven probabilistic classifier models. Advantage of this system is that we can basically plug in any classifier model that will give us prediction of the class with the corresponding probability, thus we have have also performed experiments with several models of this category.

We have also introduced the evaluation framework for our mechanisms based on this probabilistic classifier model. We have performed all the evaluations based on this framework, that were possible to perform on our available data.

First of all we have evaluated that our driver response models works and we are able to learn some patterns from the available data. On the other hand we can say that our models are not absolutely perfect. This might be caused by several factors. First of all we will probably never be able to record all the variables that describes whole passenger taxi matchmaking mechanism. From this point of view we can name factors like weather or some subjective feeling of either of sides. Last example bring us to other problem which is a human factor which is unfortunately on the both sides of our mechanism. It's usually very difficult to predict very specific human behavior related situations. As an example we can name that driver might see someone on the street at the same time when he was broadcasted by the application and naturally his optimal behavior would be to give a ride to the hailing passenger, but there might be also others very hard to predict situation like driver falling asleep during the waiting, driver having a puncture or many others.

On the other hand when we were performing evaluation of the whole mechanism that contained our model we could have seen that our mechanism produces reasonable results, but we can just hardly tell if our system would really perform better than the already existing mechanism that uses few basic heuristics. One from the reasons why our evaluations did not clearly answered this question were some inconsistencies in the datasets that we had available. On the other hand we can say that further testing of our mechanism can be quiet easily done aside of the currently working mechanism. So the first step towards implement-

ing our system in production that we recommend is to perform once again our matchmaking mechanism evaluation aside of the real world system. After this experiment there might be performed first implementations of our mechanism to the real world system.

In terms of particular machine learning models comparison we can see that the best performing model for us was the decision tree forest. We could have seen that until we have used only several features available directly from the data Naive Bayes worked quiet well for us, but unfortunately after computing another relevant features from the data we could have seen performance deterioration of the Naive Bayes, but also the improvement of the decision tree forest. This fact makes also decision tree forest promising algorithm towards introducing another new relevant features that might appear together with some additional recorded fields or other changes of the currently working system.

7.1 Future work

There are still remaining several ideas of what can be possibly improved and investigated in order to even improve our mechanism.

First of all we can see huge potential in usage of the deep learning methods instead of the standard machine learning methods that we have used. This might be advantageous for two reasons. First of all there might be performed some kind of automatic feature identification and creation using deep learning. This might even simplify our work and might be even better for future adding of new data fields. Second advantage of deep learning might be even in the improvement of the machine learning model prediction performance. Someone might argue that we might not have available enough data for the usage of the deep learning, but there exists several tricks how to work with datasets that contain tens to hundreds thousands records same as our dataset. We can see that several datasets published at deeplearning.net[1] has the comparable size as our training dataset, same as several Kaggle competitions[2] that were won by the deep learning methods.

Our other idea of what can be possibly done in the future to improve our mechanism is the combination of our model together with the data about passengers and drivers behavior in the applications. From these maybe little bit more business intelligence point of view models we could for example extract several new input features for our models. These features might for example represent passengers and drivers better than using simple K-means on the few data fields that are available for us from the transactions dataset.

Another field that might improve the matchmaking mechanism a lot is the own design of the mechanism. In our work we were pretty much fixed to the currently working mechanism, but introducing some other changes to the mechanism from extending the time that drivers has for the reply to some motivation for the drivers to accept more request might also lead to the improvement of the existing system.

[1] <http://deeplearning.net/datasets/>
[2] <http://benanne.github.io/2014/04/05/galaxy-zoo.html>

Bibliography

[1] Network Performance Traffic Analysis Centre *Traffic levels on major roads in Greater London*, Published by Transport for London in March 2012, available at <http://www.tfl.gov.uk/cdn/static/cms/documents/traffic-note-1-traffic-levels-in-greater-london-2010.pdf>

[2] Ryan Jay Herring *Real-Time Modeling and Estimation with Streaming Probe Data using Machine Learning*, Published at eSholarship University of California in Fall 2010, available at <http://bayen.eecs.berkeley.edu/sites/default/files/thesis/ryh10.pdf>

[3] Carlos Gershenson *Living in Living Cities*, Published at arXiv.org in November 2011, available at <http://arxiv.org/pdf/1111.3659v3.pdf>

[4] Jean-François Cordeau, Gilbert Laporte *The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms*, Published at Quarterly Journal of the Belgian, French and Italian Operations Research Societies 1.2 (2003) in June 2002, available at <http://www.dim.uchile.cl/~tcapelle/BIBLIOGRAFIA%20TESIS/Laporte.pdf>

[5] Rémy Chevrier, Arnaud Liefooghe, Laetitia Jourdan, Clarisse Dhaenens *Solving a Dial-a-Ride Problem with a Hybrid Evolutionary Multi-objective Approach: Application to Demand Responsive Transport*, Published at Elsevier in 2012, available at <http://hal.inria.fr/docs/00/67/85/82/PDF/chevrier_ASOC2012.pdf>

[6] Gerardo Berbeglia, Jean-Francois Cordeau, Gilbert Laporte *A Hybrid Tabu Search and and Constraint Programming Algorithm for the Dynamic Dial-a-Ride Problem*, Published at INFORMS Journal on Computing in 2012, available at <https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2010-14.pdf>

[7] M. Schilde, K.F. Doerner, R.F. Hartl *Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem*, Published at Elsevier in 2014, available at <http://www.sciencedirect.com/science/article/pii/S0377221714002197>

[8] K.Sairam, R.Nandakrishnan, Veeramuth Venkatesh *Competent Smart Car parking: An OSGi Approach*, Published in Journal of Artificial Intelligence 6.1 in 2013, available at <http://docsdrive.com/pdfs/ansinet/jai/0000/48352-48352.pdf>

[9] Yanfeng Geng, Christos G. Cassandras *New "Smart Parking" System Based on Resource Allocation and Reservations*, Published at Intelligent Transportation Systems, IEEE

Transactions on 14.3 in 2013, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6082832>

[10] Claudia Di Napoli, Dario Di Nocera, and Silvia Rossi *Agent Negotiation for Different Needs in Smart Parking Allocation*, Published by Springer International Publishing in 2014, available at <http://link.springer.com/chapter/10.1007/978-3-319-07551-8_9>

[11] Shuo-Yan Chou, Shih-Wei Lin, Chien-Chang Li *Dynamic parking negotiation and guidance using an agent-based platform*, Published at Expert Systems with Applications in 2008, available at <http://www.sciencedirect.com/science/article/pii/S095741740700293X>

[12] Soumya Banerjee, Hameed Al-Qaheri *An intelligent hybrid scheme for optimizing parking space: A Tabu metaphor and rough set based approach*, Published in Egyptian Informatics Journal in 2011, available at <http://www.sciencedirect.com/science/article/pii/S1110866511000077>

[13] David William Daddio *MAXIMIZING BICYCLE SHARING: AN EMPIRICAL ANALYSIS OF CAPITAL BIKESHARE USAGE* , Published by University of North Carolina at Chapel Hill in 2012, available at <http://rethinkcollegepark.net/blog/wp-content/uploads/2006/07/DaddioMP_Final-Draft.pdf>

[14] Oliver O'Brien, James Cheshire, Michael Batty *Mining bicycle sharing data for generating insights into sustainable transport systems*, Published in J Transport Geography in 2013, available at <http://www.complexcity.info/files/2013/08/BATTY-JTG-2013.pdf>

[15] Patrick Vogela, Torsten Greisera, Dirk Christian Mattfelda *Understanding Bike-Sharing Systems using Data Mining: Exploring Activity Patterns*, Published at Procedia-Social and Behavioral Sciences 20 in 2011, available at <http://www.sciencedirect.com/science/article/pii/S1877042811014388>

[16] G.K.D Saharidis, A. Fragkogios and E. Zygouri *A Multi-Periodic Optimization Modeling Approach for the Establishment of a Bike Sharing Network: a Case Study of the City of Athens*, Published at Proceedings of the International MultiConference of Engineers and Computer Scientists in 2014, available at <http://www.iaeng.org/publication/IMECS2014/IMECS2014_pp1226-1231.pdf>

[17] Juan P. Romero, Angel Ibeas, Jose L. Moura, Juan Benavente, Borja Alonso *A simulation-optimization approach to design efficient systems of bike-sharing*, Published at Procedia-Social and Behavioral Sciences in 2012, available at <http://www.sciencedirect.com/science/article/pii/S1877042812042449>

[18] Pierre Borgnat, Céline Robardet, Jean-Baptiste Rouquier, Patrice Abry, Eric Fleury, and Patrick Flandrin *SHARED BICYCLES IN A CITY: A SIGNAL PROCESSING AND DATA ANALYSIS PERSPECTIVE*, Published at Advances in Complex in 2011, Systems available at <http://hal.archives-ouvertes.fr/docs/00/49/03/25/PDF/velov_acs.pdf>

[19] Julius Pfrommer, Joseph Warrington, Georg Shildbach, Manfred Morari *Dynamic vehicle redistribution and online price incentives in shared mobility systems*, Published at arXiv.org in 2013, available at <http://arxiv.org/pdf/1304.3949.pdf>

[20] Karama Jeribi, Hinda Mejri, Hayfa Zgaya, Slim Hammadi *Vehicle Sharing Services Optimization Based on Multi-Agent Approach*, Published at 18th World Congress of the International Federation of Automatic Control in 2011, available at <http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac11-proceedings/data/html/papers/0615.pdf>

[21] Jia Shu, Mabel Chou, Qizhang Liu, Chung-Piaw Teo, I-Lin Wang *Bicycle-Sharing System: Deployment, Utilization and the Value of Re-distribution*, Published by National University of Singapore-NUS Business School, Singapore in 2010, available at <http://bschool.nus.edu/Staff/bizteocp/BS2010.pdf>

[22] Andrey Glaschenko, Anton Ivaschenko, George Rzevski, Petr Skobelev *Multi-Agent Real Time Scheduling System for Taxi Companies*, Published at 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest in 2009, available at <http://www.ifaamas.org/Proceedings/aamas09/pdf/03_Industrial_Track/13_70_it.pdf>

[23] Kiam Tian Seow, Nam Hai Dang and Der-Horng Lee *A Collaborative Multiagent Taxi-Dispatch System* , Published at Automation Science and Engineering, IEEE Transactions in 2010, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5286312>

[24] Jan Zikeš *Auction-based Taxi Allocation with Dynamic Pricing*, published at CTU in 2012, available at <https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13136&y=2012&a=zikesjan&t=bach>

[25] Thuy T.T. Nguyen and Grenville Armitage *A Survey of Techniques for Internet Traffic Classification using Machine Learning*, Published in Communications Surveys & Tutorials, IEEE in 2008, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4738466>

[26] Shiliang Sun, Changshui Zhang, Guoqiang Yu *A bayesian network approach to traffic flow forecasting*, Published at Intelligent Transportation Systems, IEEE Transactions in 2006 available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1603558>

[27] AiLing Ding, XiangMo Zhao, Licheng Jiao *Traffic flow time series prediction based on statistics learning theory*, Published at The IEEE 5th International Conference on. IEEE in 2002, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1041308>

[28] Bin Li, Daqing Zhang, Lin Sun, Chao Chen, Shijian Li, Guande Qi, Qiang Yang *Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset*, Published at IEEE International Conference in 2011, available at <http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=5766967>

[29] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, Guangzhong Sun *Where to Find My Next Passenger?*, Published at Proceedings of the 13th international conference on Ubiquitous computing in 2011, available at <http://dl.acm.org/citation.cfm?id=2030128>

[30] Moreira-Matias L., Gama J., Ferreira M., Mendes-Moreira J., Damas, L. *Predicting Taxi–Passenger Demand Using Streaming Data*, Published at ieeexplore.ieee.org in 2013, available at <http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?tp=&arnumber=6532415>

[31] Santi Phithakkitnukoon, Marco Veloso, Carlos Bento, Assaf Biderman, Carlo Ratti *Taxi-Aware Map: Identifying and Predicting Vacant Taxis in the City*, Published by Springer Berlin Heidelberg in 2010, available at <http://link.springer.com/chapter/10.1007/978-3-642-16917-5_9>

[32] Yuan, N.J., Yu Zheng, Liuhang Zhang, Xing Xie *T-Finder: A Recommender System for Finding Passengers and Vacant Taxis*, Published at Knowledge and Data Engineering, IEEE Transactions in October 2010, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6261314>

[33] Yang Yue, Yan Zhuang, Qingquan Li, Qingzhou Mao *Mining Time-dependent Attractive Areas and Movement Patterns from Taxi Trajectory Data*, Published at 17th International Conference on. IEEE in 2009, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5293469>

[34] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, Yan Huang *T-Drive: Driving Directions Based on Taxi Trajectories*, Published at 18th SIGSPATIAL International conference on advances in geographic information systems in 2010, available at <http://dl.acm.org/citation.cfm?id=1869807>

[35] Alexandru Niculescu-Mizil, Rich Caruana *Predicting Good Probabilities With Supervised Learning*, Published at 22nd international conference on Machine learning in 2005, available at <http://machinelearning.wustl.edu/mlpapers/paper_files/icml2005_Niculescu-MizilC05.pdf>

[36] Ashutosh Garg and Dan Roth *Understanding Probabilistic Classifiers*, Published by Springer Berlin Heidelberg in 2002, available at <http://l2r.cs.uiuc.edu/~danr/Papers/ecml01.pdf>

[37] Leo Breiman *Random forests*, Published in Machine learning 45.1 in 2001, available at <http://link.springer.com/article/10.1023/A:1010933404324>

[38] Guoqiang Peter Zhang *Neural Networks for Classification: A Survey*, Published in Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions in 200, available at <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=897072>

[39] John C. Platt *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods (1999)* available at <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1639&g>

[40] Amir F. Atiya *Estimating the Posterior Probabilities Using the K-Nearest Neighbor Rule*, Published in Neural computation in 2005, available at <http://alumnus.caltech.edu/~amir/posterior-prob-est.pdf>

[41] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, Hang Li *Learning to rank: from pairwise approach to listwise approach*, Published at 24th international conference on Machine learning in 2007 available at <http://dl.acm.org/citation.cfm?id=1273513>

[42] Fen Xia, Tie-Yan Liu, Hang Li *Top-k Consistency of Learning to Rank Methods*, Published at Advances in Neural Information Processing Systems in 2009, available at <http://research.microsoft.com/pubs/103073/topk-tr.pdf>

[43] Stephen Marsland *Machine Learning: An Algorithmic Perspective*, Published by Chapman & Hall/CRC in 2009

[44] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V. Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E. *Scikit-learn: Machine Learning in Python*, Published at The Journal of Machine Learning Research in 2011, available at <http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>

Appendix A

Data analysis

We were provided the data that describes two step passenger taxi matching in the real world. In the first step we are trying to determine the probability of taxi driver accepting, rejecting or letting be time outed the passengers request. We have tried to visualize several "prior" probabilities with respect to various features from data.

All the visualizations were made separately for whole data set and for all the data from last 28 days.

A.1 Visualized features in the first step

In this step we are interested only in that the taxi driver will accept the passengers request we don't care about that if the passengers acceptance of the journey will follow.

A.1.1 Prior probability based on the day in a week

We have observed that this prior probability differs among particular days of the week. We were able to observe that the lowest prior probability of acceptance was measured on Saturday, approximately 0.21 and the highest on Wednesday, approximately 0.27. When we have looked only on the data from last 28 days we were able to see more or less the similar pattern of the drivers behavior. For details see figure A.1.

A.1.2 Prior probability based on the hour in a day

From our visualizations it is possible to see that the highest prior probability of request being accepted by the taxi driver is between 23:00 and 24:00 around 0.33 and the lowest prior probability is between 16:00 and 17:00 around 0.17. When we have visualized the same for last 28 days we could see that the highest prior probability of request being accepted by the taxi driver is between 21:00 and 22:00 and then between 23:00 and 24:00 this probability is around 0.31. On the other hand the worst situation is between 16:00 and 17:00 when the prior probability is only 0.13. For details see figure A.2.

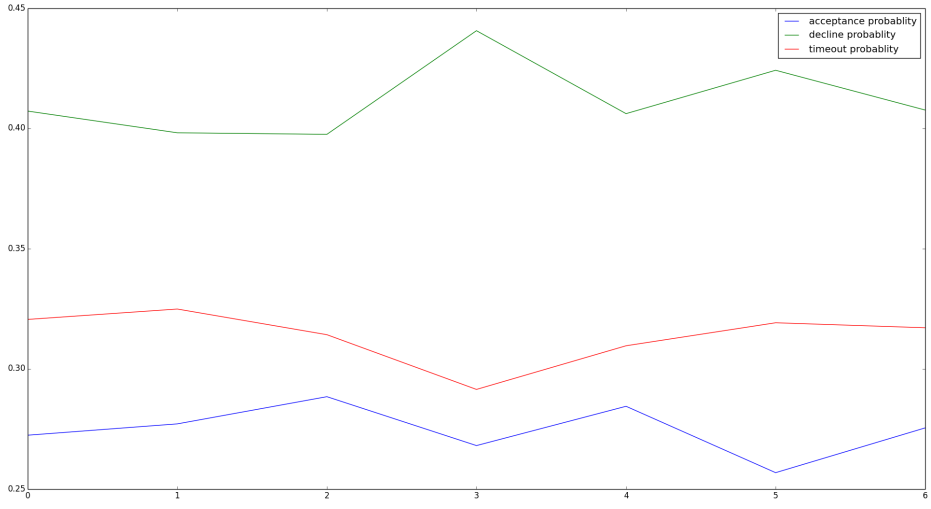


Figure A.1: step1: prior probability based on the weekday

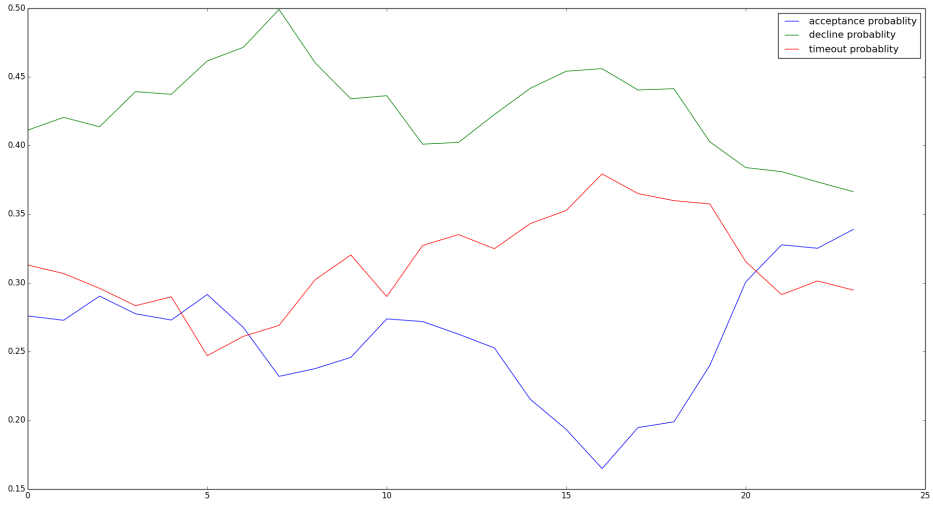


Figure A.2: step1: prior probability based on hour of the day

A.1.3 Prior probability based on both time of the day and day in a week

We can see that the highest request acceptance prior probability is during the workdays around midnight, it is usually from 0.3 to 0.4 and by far the lowest probability probability of the taxi request being accepted we can see on Friday between 16:00 and 17:00 when the probability is only 0.1. When we have look on the visualization of the data for last 28 days we can see that the highest prior probability of taxi request will be accepted we have in nights except night from Friday to Saturday and from Saturday to Sunday, again the number varies from 0.3 to 0.4. On the other hand the lowest prior probability we can expect on everyday afternoon except Sunday, this probability is usually again around 0.1. For details see figure A.3.

A.1.4 Prior probability based on the distance between passenger and taxi.

We can see that the highest prior probability of request being accepted by the taxi driver we have when the distance between taxi driver and passenger is small in this case it is around 0.5 then it decreases to values around 0.2 when the distance is from 6 to 10 kilometers. In last 28 days we were able to see again more or less the same pattern. More details can be seen on figure A.4.

A.1.5 Prior probability based on the estimated traveling time of the taxi driver to the passenger

We can see that as we have expected there is the highest prior probability when the traveling time is low, around 0.45 and low when the traveling time is higher, around 0.1. We can also see more or less the same pattern when we have look only on the data from last 28 days. More details can be seen on figure A.5.

A.1.6 Drivers acceptance prior probability with respect to the drivers geographical position

We can see that there is slightly higher probability of the passengers request being accepted when the driver is in the city center and it decreases towards the suburbs. When we visualize data from the last 28 days we can see that this pattern is even more significant. More details you can see on figure A.6.

A.1.7 Drivers acceptance prior probability with respect to the passengers geographical position

From our visualizations there is possible to see that there is higher probability for passengers request to be accepted when the passenger is closer to the city center and lower when he is in suburbs. When we have look on data for last 28 days than it seems that there are several spots where the prior acceptance probability is higher and it seems that this time it does not directly corresponds to the city center. More details you can see on figure A.7.

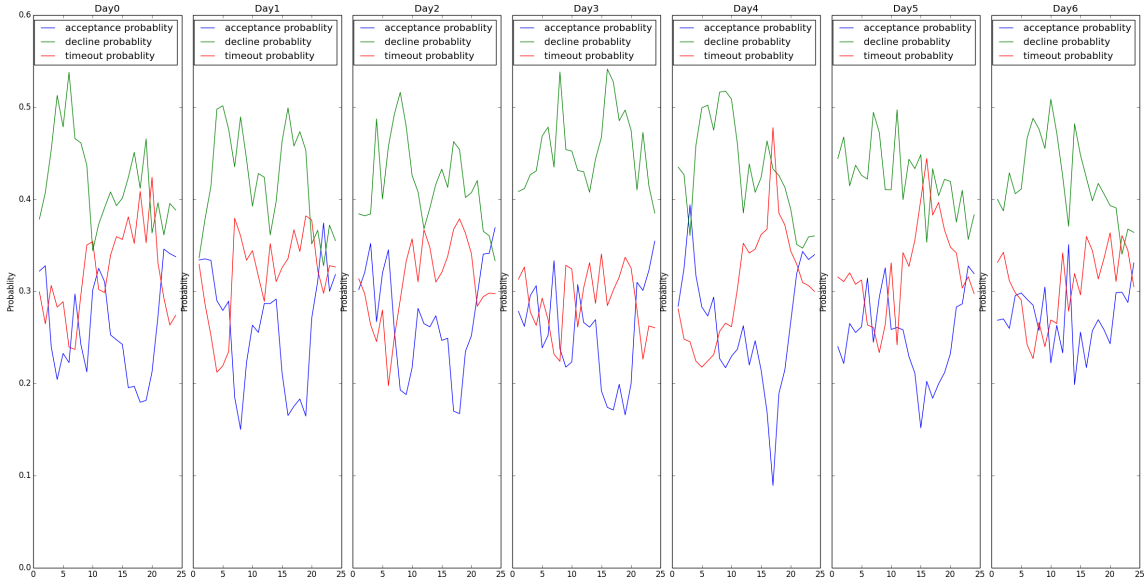


Figure A.3: step1: prior probability based on hour of the day and day in a week

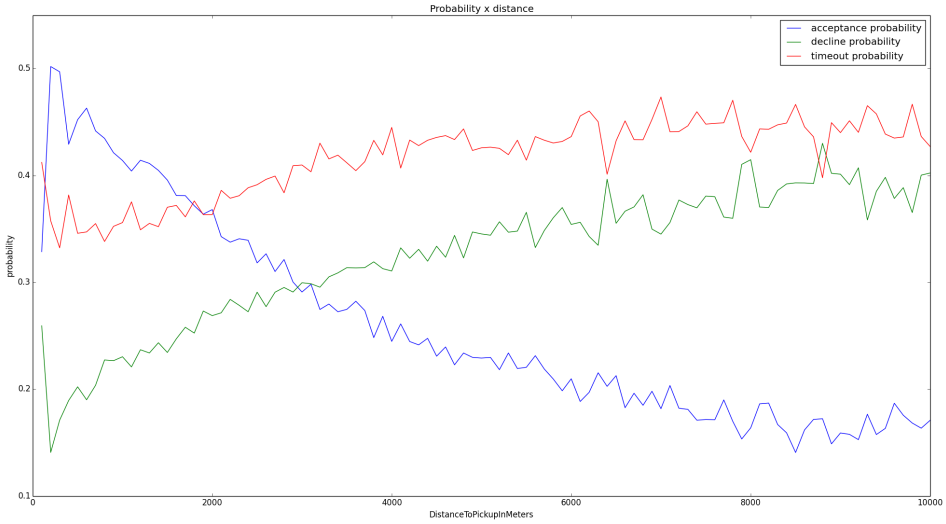


Figure A.4: step1: prior probability based on distance between driver and passenger

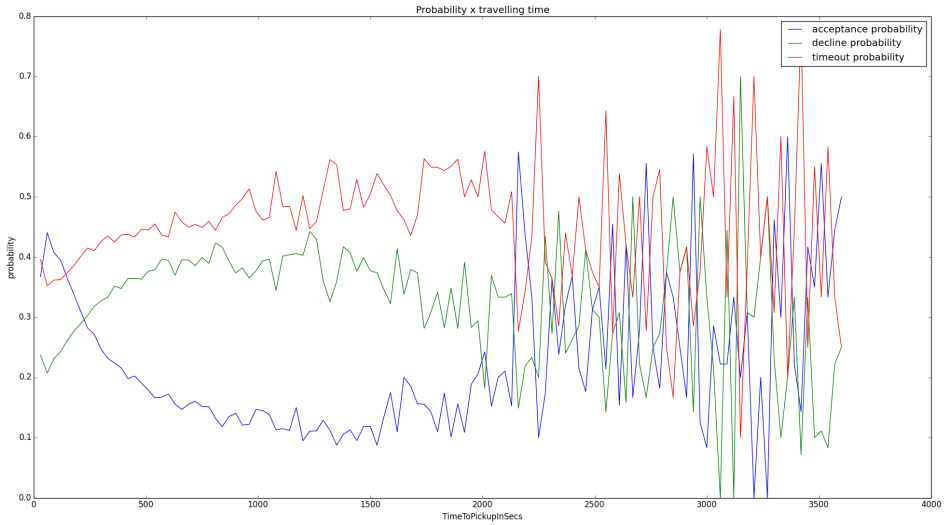


Figure A.5: step1: prior probability based on traveling time between driver and passenger

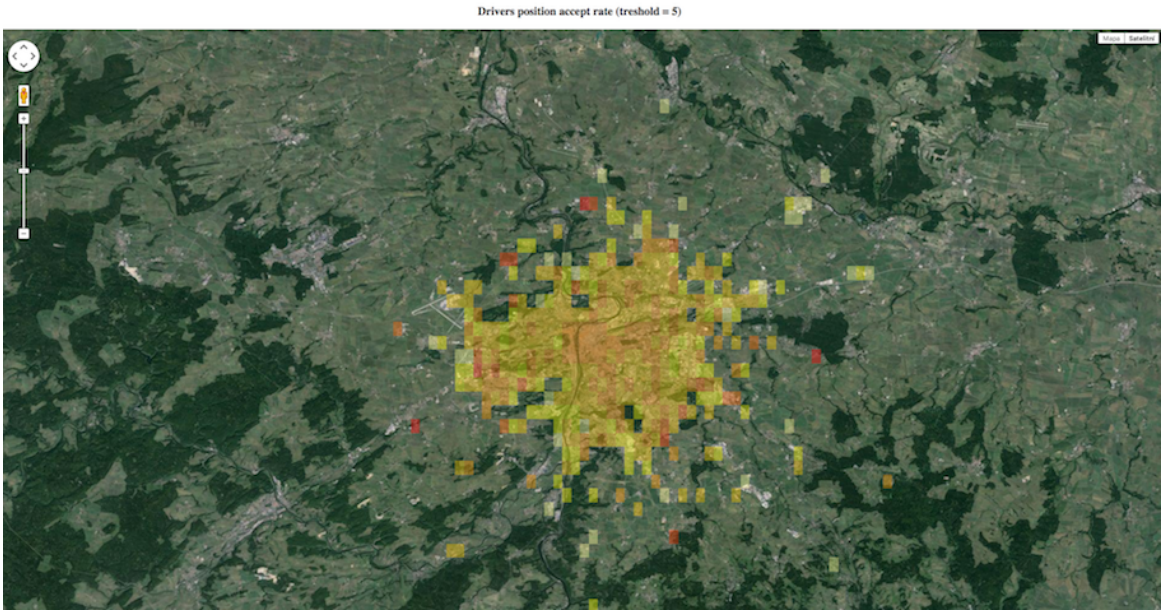


Figure A.6: step1: prior probability based on the taxi drivers geographical position

A.1.8 Drivers acceptance prior probability with respect to that if the destination was entered

We can see that there is slightly higher prior probability of trip request being accepted by the taxi driver when there was entered the destination. Particularly prior probability without the destination entered was 0.24 and with the destination entered 0.28. On the other hand when we have a look on the data from last 28 days we can see that the probability of acceptance without entered destination was 0.23 and with entered destination 0.24. Details can be seen on figure A.8.

A.1.9 Conclusion

From our visualizations we can see that we were able to see that values of our features directly correlates with the request being accepted by the taxi driver. On the other hand we can't clearly say that our measured features are independent, thus we can not only multiply these prior probabilities to get our final predicted probability of request being accepted by the taxi driver. One example of features that are clearly not independent we can name distance between taxi driver and passenger and traveling time between passenger and taxi driver.

A.2 Visualized features in the second step

In this step we are interest in that the whole journeys will be successfully set up after the request was accepted by the taxi driver.

A.2.1 Prior probability based on the traveling time of the taxi driver to the passenger

We can see that again there is higher prior probability that the passenger will accept drivers offer when the traveling time of the driver to the passenger is low, particularly around 0.43 and it gets lower to 0 when the traveling time is high. When we have visualized data for last 28 days we have seen that the pattern starts in a same way, but then from certain traveling time we don't have enough data and the data we have are basically only noise. More details you can see on figure A.9.

A.2.2 Prior probability based on the distance between the taxi driver and the passenger

Quiet similar observation as in the section of traveling time to the passenger we can also get from the plot of prior probabilities with respect to the distance between passenger and the taxi driver. The prior probability of acceptance when the passenger was close to the taxi was more than 0.5 and when the passenger was far than it has decreased to less than 0.2. In the data from last 28 days we were able to see that the decrease of the prior probability with distance was not that steep, but unfortunately from certain values we did not have enough data and in the visualization we can see basically only noise. More details you can see on figure A.10.

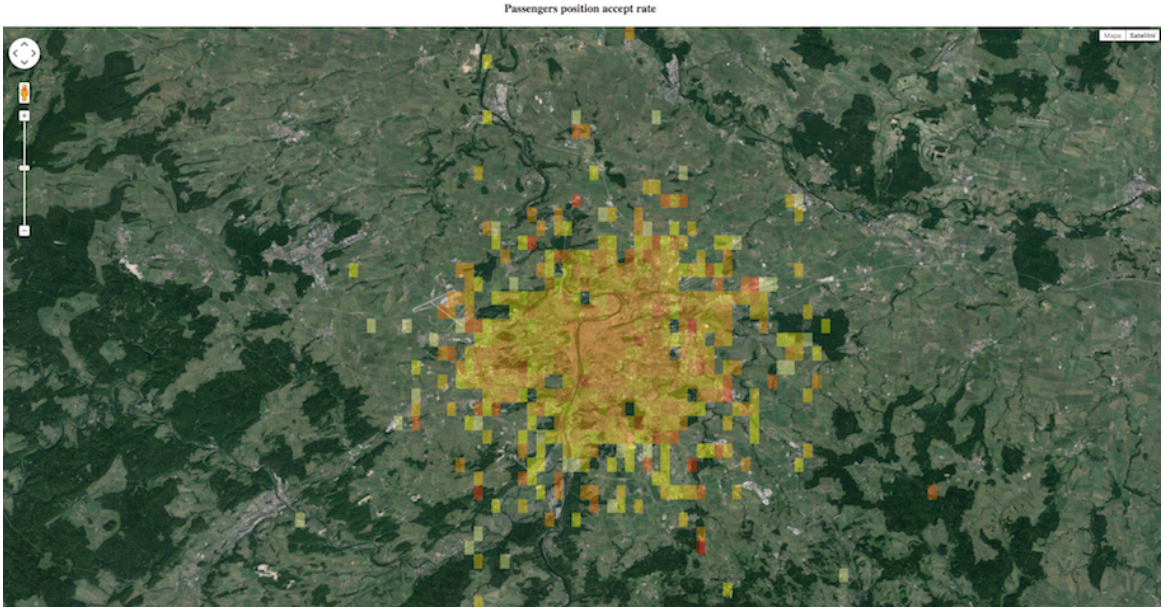


Figure A.7: step1: prior probability based on the passengers geographical position

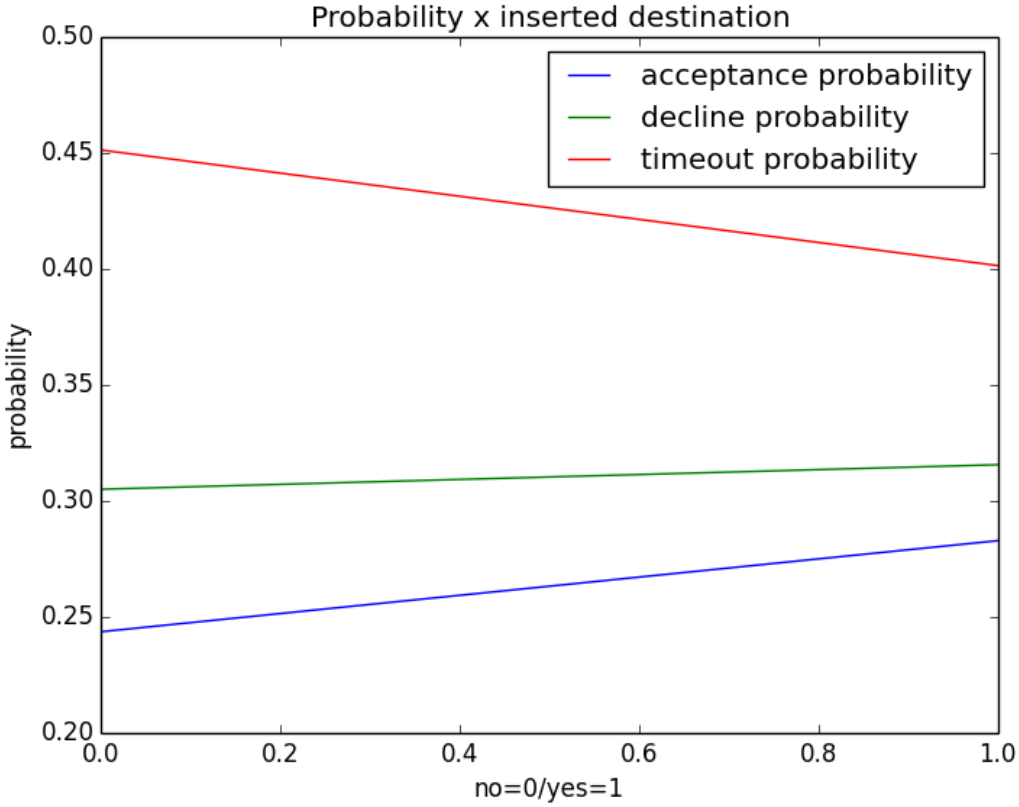


Figure A.8: step1: prior probability based on if the destination was entered

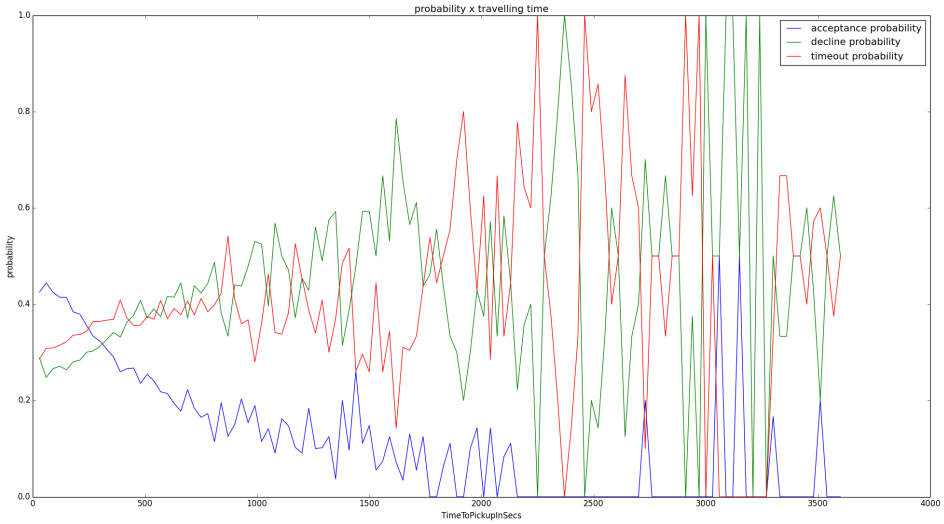


Figure A.9: step2: prior probability based on if the traveling time between passenger and driver

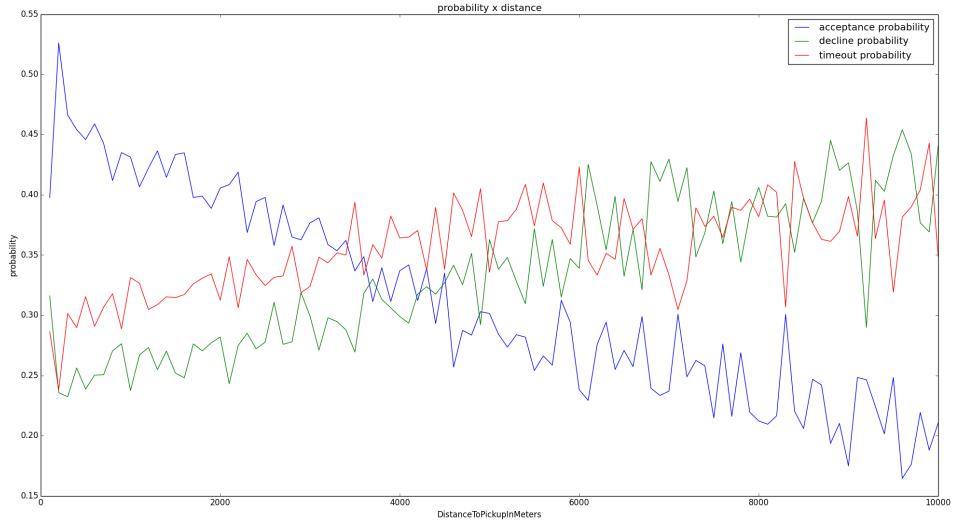


Figure A.10: step2: prior probability based on if the distance between passenger and driver

A.2.3 Prior probability based on the price per km quoted by the taxi

We can see that in the price range where are the most of taxis in Prague, between 13CZK and 28CZK there is possible to see that prior probability slightly decreases as the price increases. On the other hand in the data from last 28 days we can see that there is almost no correlation between price quoted by the taxi and prior probability of that the proposed journey will be accepted by the passenger. More details you can see on figure A.11.

A.2.4 Prior probability based on the hour in a day

From the plots it is possible to see that the highest prior probability of taxi drivers offer being accepted by the passenger is in the night hours, especially from 4:00 to 5:00 particularly 0.45. On the other hand the lowest probability of accepting the offer from the taxi drivers is between 12:00 and 13:00 and then between 17:00 and 18:00 when it is 0.25. When we have look at the data from last 28 days we can see that the situation was completely different. The highest probability of the offer being accepted from the passenger we can see between 10:00 and 13:00 when it is almost 0.55 and the lowest, slightly around 0.35 in the morning from 8:00 to 9:00 and then from 15:00 to 17:00. For more details see figure A.12.

A.2.5 Prior probability based on the day in a week

From our visualizations there is possible to see that the lowest prior probability of taxi drivers offer being accepted by the passenger is on Monday, slightly over 0.32 on the other hand the highest probability we can see on Saturday, particularly slightly above 0.36. When we have look only on the data from last 28 days we can see that the situation has changed and the lowest probability is on Saturday, around 0.34 and the highest on Wednesday, approximately 0.38. For more details see figure A.13.

A.2.6 Prior probability based on both the weekday and the time of a day

We can see That the highest prior probabilities of taxi drivers offer being accepted by the passenger is in nights from Monday to Tuesday, from Tuesday to Wednesday, from Wednesday to Thursday and from Thursday to Friday when the highest probability is usually slightly after the midnight and it is around 0.55 on the other hand the lowest prior probability we can expect on Sunday in the afternoon when it is around 0.15 and in the Monday between 4:00 and 5:00 when it is 0.18. When we have look on the data from the last 28 days we can clearly see that we don't have enough data for some conclusion. For more details see figure A.14.

A.2.7 Prior probabilities based on the passengers device

We have also performed visualizations separately for the passengers that are using device with android and those who are using iOS. We were able to see that there is slightly higher probability of accepting the offer by the iOS users and there is also seen slightly higher prior probability of the iOS user to accept more expensive driver and driver that travels from the bigger distance.

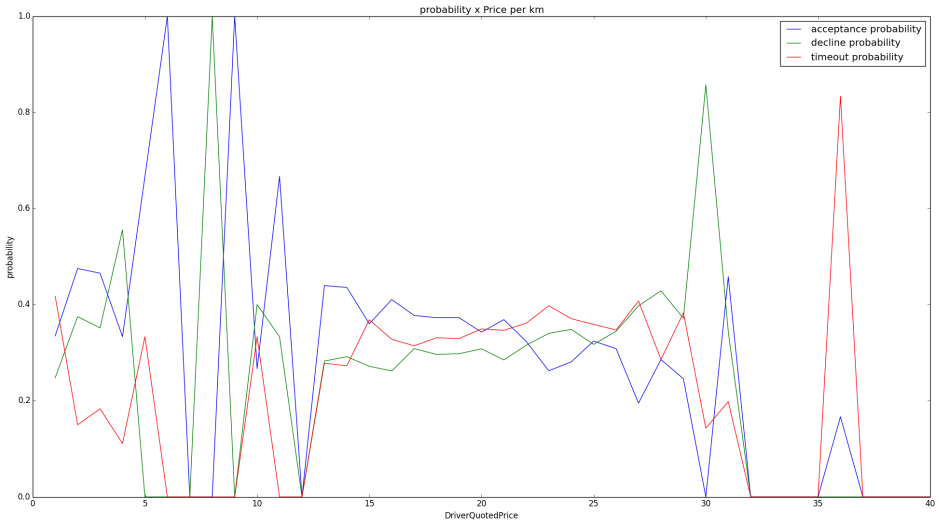


Figure A.11: step2: prior probability based on the drivers quoted price

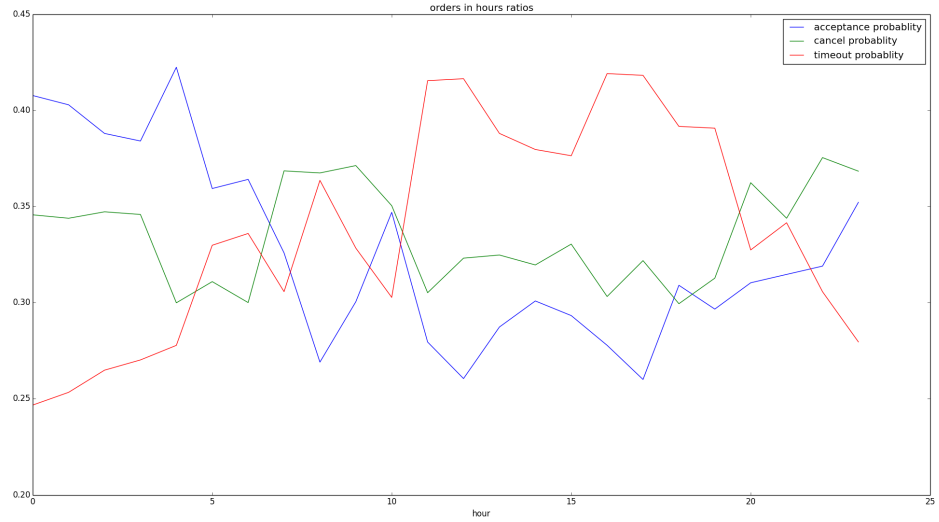


Figure A.12: step2: prior probability based on hour in a day

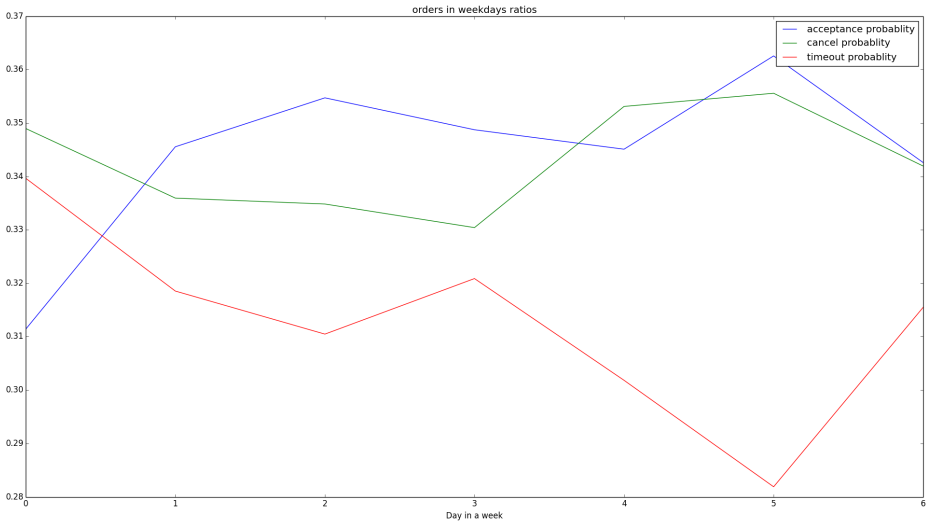


Figure A.13: step2: prior probability based on the day in a week

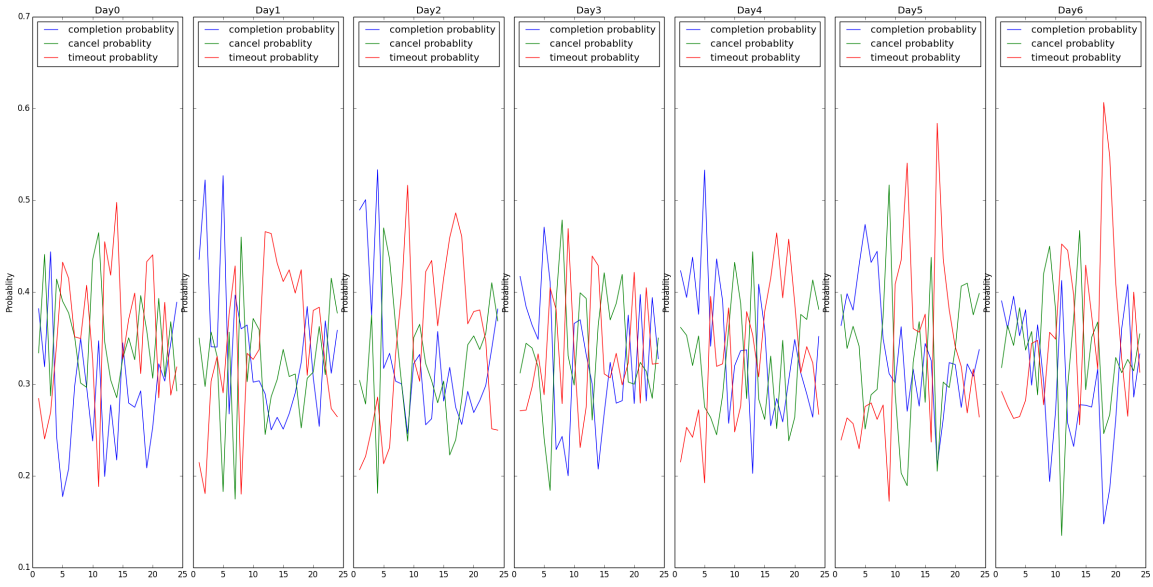


Figure A.14: step2: prior probability based on both the day in a week and hour in a day

A.2.8 Prior probability based on the drivers geographical position

From the visualizations we can see that generally the prior probability of taxi being accepted is high in the city center, but then it is also high at some specific locations in the suburbs. In the data from last 28 days we can see that prior probability of driver being accepted by the passenger is high at some suburban spots. For more details see map on figure A.15.

A.2.9 Prior probability based on the passengers geographical position

From the data we can see that there are clearly locations from where passengers prior probability of accepting the taxi driver is higher than in others, but we can't generally say this happens in the center or in the suburbs. Exactly the same situation we can see from the data from last 28 days. For more details see map on figure A.16.

A.2.10 Conclusion

We can several correlations between particular recorded features and the probability of the taxi drivers offer being accepted. On the other hand we can also see several dependencies between particular features, again as a example we can name taxi geographical positions of both taxi drivers and passengers with distances and traveling time.

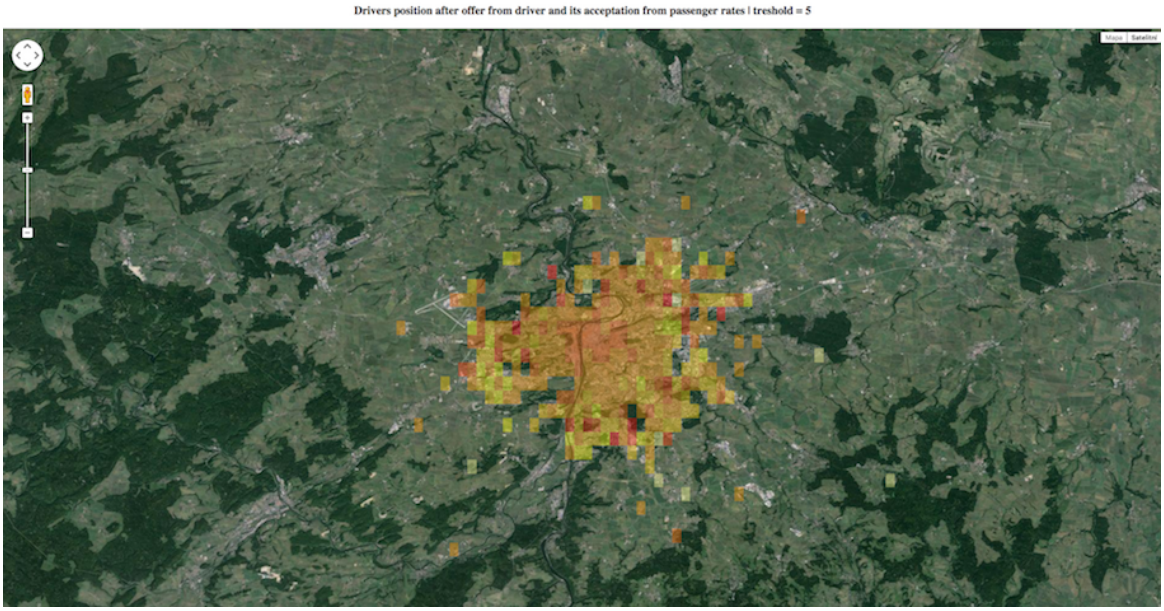


Figure A.15: step2: prior probability based on the geographical position of the taxi driver

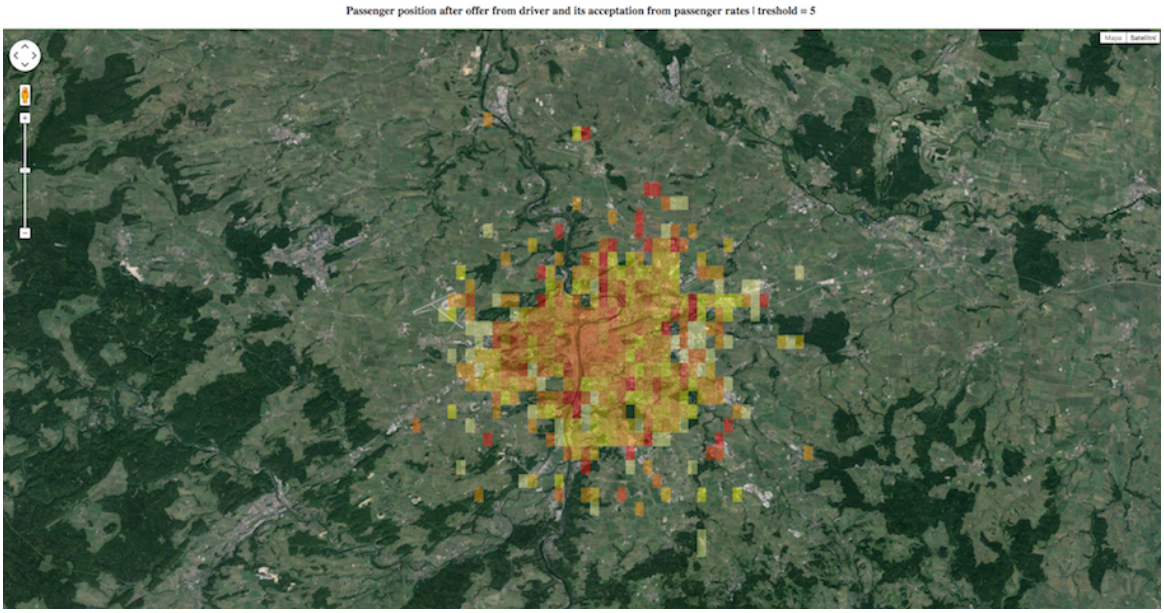


Figure A.16: step2: prior probability based on the geographical position of the passenger

Appendix B

User Guide

Before you try to run the program from the enclosed CD you should read following few information about how to make the program run and where and which variables is possible to set.

System Requirements

Program should be possible run on the most of the major operating systems. Though installed Python2.7 and compatibile version of LAPACK is needed.

Dependencies

To make the source code from the enclosed CD work you have to first download some publicly available dependencies. The dependencies can be easily installed using pip. Particularly by running the command "pip install -r requirements.txt" in the code root directory. In case that you are for some reason not using pip you can also manually install all of the packages separately. For all the needed packages please see requirements.txt.

Variables

Geo utils variables

In the script geoutils.py you can experiment with various variables, especially with the Prague bounding box or with the heuristically set center of Prague.

Model Variables

In the script drivers_models.py you can experiment with different parameters of the particular models as it was described in the section 6.3.

Preprocessing cariables

In the script feature_builder.py you can also experiment with different K representing the number of clusters used for feature building.

Appendix C

Content of the CD

```
.
├── DP.pdf                          -Document with the diploma thesis
├── Python_scripts                  -Folder containing the Python
scripts
│   ├── data_preprocessor.py        -Data preprocessor script
│   ├── drivers_models.py           -Drivers model script
│   ├── evaluator.py                -Evaluator script
│   ├── feature_builder.py          -Feature builder script
│   ├── feature_extractor.py        -Feature extractor script
│   ├── geoutils.py                 -Geoutils script
│   ├── __init__.py
│   ├── requirements.txt            -Requirements file for pip
│   └── trees_visualizer.py         -Tree visualise script
└── tex_sources                     -Tex sources of the Diploma thesis
    ├── DP_text.aux
    ├── DP_text.lof
    ├── DP_text.log
    ├── DP_text.lot
    ├── DP_text.out
    ├── DP_text.pdf
    ├── DP_text.synctex.gz
    ├── DP_text.tex                 -Tex file with the Diploma thesis
    ├── DP_text.toc
    ├── figures                     -Folder containing figures
    ├── hyphen.tex
    ├── k336_thesis_macros.sty
    └── Makefile
```

Figure C.1: Content of the enclosed CD