RESEARCH PROJECT REPORT

by

Venkata Kolagotla(930096)

Supervisor : Prof. Dr. Jens Lüssem

Outline

Abstract

In this report the task of analyzing the Flinkster booking data as part of a research project is being performed. The main objective of this project was to get an idea about how the data analysis is usually been performed in a real time company and also make use of infrastructure in the university. Given the size of the data, the use of infrastructure in the university was not needed, a local machine was able to finish the project. While doing the data analysis different time series analysis techniques were used to perform the task, some of them are know and some of them are totally new to the knowledge. After the learning process different models were built to test which one of them was best suitable for the given data. The project was maintained using a Github repository named Db-Data-Analysis. The project repository was updated at least once in a week or so throughout the project period to maintain the continuity in the project development process and learning process. All the steps in the project phase are summarized properly in the paper.

Introduction

With the growing competition in car sharing industry from the existing competitors and the new competition from bike sharing companies who are pushing the limits of the vehicle renting industry it is important to understand how the company is doing in recent times. This task has become easy with the help of the emerging technologies in data analysis industry.

The task for this project can be divided into parts.

- To identify a question which can have an impact on the business and which can be solved from the data provided, because the data was given with any particular requirement so further data analysis and interpretation was needed to identify the question by analyzing the data.
- To identify if there is sufficient data to answer the question? Or is there any other data which can help to solve the question.
- Deciding the parameters or labels to consider for the analysis, which will help solving the question.

In the Implementation section, the implementation of all these parts of the tasks is performed. This section also covers the different models used to achieve the answers and some of

the models which are used for the learning purpose. In the conclusion the final results of the project are summarized. All the refereed materials are sited in the literature section.

The total code of the project can be found in Github: https://github.com/jawaharreddy/Db-Data-Analysis

Implementation

The initial data analysis has been performed to know more about the data more. Since there was no specific requirement for the data, I had to see the data and find if there are any interesting patterns in the data. This was one of the most difficult challenges faced in the beginning of the project.

Defining the questions to solve

After the initial analysis there was a clear trend that the number of bookings per are getting reduced year by year. Based on this results the questions were formed like this?

1) Find the number of bookings happening on a in every city.

2) Find the number of cars been used on a single day in each city.

Data preparation and pivoting

The data has been acquired from the DB-Flinkster website which was already well structured. Since the data was already well structured there was no need of any data cleaning techniques to use. Although the data was well structured there was some null values in the data. The null values were in the columns DISTANCE and TECHNICAL_INCOME_CHANNEL. For the initial analysis the null values were removed and the analysis was performed.

The data has been sorted out using pivot_table package in the python pandas library. The pivot table makes it easy to filter the data and apply aggregation functions easily on the data when you have numeric data. This can be done without the pivot_table package, by using just pandas and numpy.

To get different perspective of the data given, additional data sets has been used. One of them is the population of the city and the other one is holidays list in different cities.

The data is from June 2013 till June 2017, with all the bookings happened in 2013 are being booked for 2014. To analyze the data in yearly period the there are two data frames were created. One named ‘year’ which includes 3 complete years data from 2014 till 2016, other one is ‘data’

which contains all the original data. The models which were designed and used in this project will be using the complete years dataframe which is ‘year’.

Initial data analysis

Below are some of the insights from initial data analysis:

– Most number of bookings were are happening from the official website and then IOS devices, which in turn refers that many of the bookings are maybe from business users.

```
Internet                    322260
Flinkster iPhone            101206
Flinkster Android            52171
UNKNOWN                      50197
ICS-Server                    3175
Flinkster Windows             2185
Multicity iPhone              1967
Broker HAL                    1851
Multicity Android             1407
BwFPS Portal Web              1175
BwCarsharing iPhone            872
Bahn_de_2                      811
HALAPI Teilauto                731
BwCarsharing Android           647
Book-n-Drive iPhone            544
Onesto_Bahn                    336
API                            236
```

Fig1: Number of bookings from different sources.

– The cities with high population have more number of customers, more number of bookings and more number of cars, except some of exceptions like Hamburg witch has high population but the number of bookings and number of customers are less compared to other big cities.

		BOOKING_HAL_ID	CUSTOMER_HAL_ID	START_RENTAL_ZONE	VEHICLE_HAL_ID
CITY_RENTAL_ZONE	POPULATION				
Berlin	**3520031**	153768	15669	113	526
München	**1450381**	87216	8255	58	288
Köln	**1060582**	96114	8002	72	252
Stuttgart	**623738**	37682	4944	37	150
Hamburg	**1787408**	19675	4183	9	78
Frankfurt am Main	**732688**	17948	4148	10	64
Hannover	**532163**	9001	2283	2	36
Mannheim	**305780**	10347	2218	4	37
Freiburg	**226393**	9640	2018	1	38
Karlsruhe	**307755**	7154	1555	3	27

Fig2: Comparison between cities with high population

– The number of bookings are getting decreased at the end of the year due to the long holidays at the end of the year.

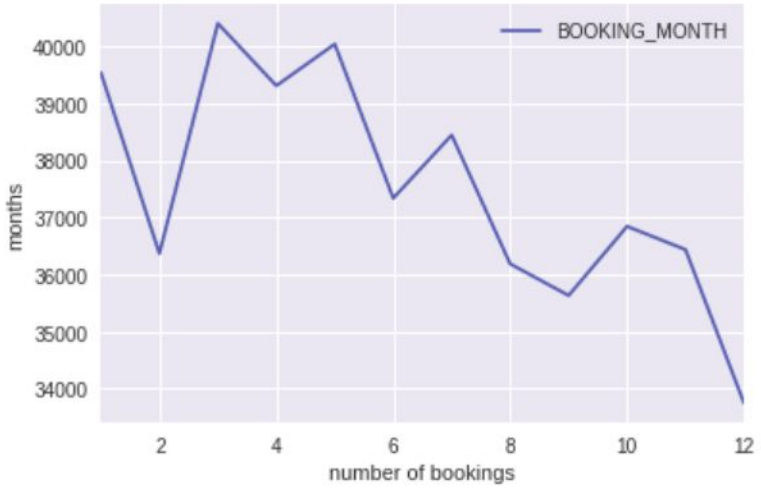

Fig3: Booking pattern over monthly data

Deciding which labels to consider

Once the questions were finalized it became easy to find the corresponding labels which are important to answer these questions. To find the number of bookings, the columns BOOKING_HAL_ID, DATE_BOOKING were considered and to find the number of cars used, the columns DATE_BOOKING, VEHICLE_HAL_ID were considered. At this stage the the columns with null values are also considered since the null values had no effect on the analysis.

The label DATE_FROM is also considered in the later stages of the project with using this label to find the number of cars used made more sense. The DATE_FROM tell the number cars used on particular day whereas DATE_BOOKING gives the information about the number cars used on a where there might be other bookings happened for different date, which would mean that the car is used on different day even though the booking happened on the current day.

Time series analysis

After the initial data analysis it has been decided that the best way to answer the questions is by doing time series analysis on the given data. The data is booking information from one particular period of time to another time period, so this was best suited model.

A time series data is collection of a value collected over different and mostly at regular time intervals. One main difference between standard linear regression problem and time series problem is that the data is not necessarily independent and not necessarily identically distributed. One characteristic of time series data is that this is a list of observations where the order of the data matters. Ordering of this data is very important because there is dependency in the data and changing this order could change the meaning of the whole data.

Choosing the appropriate model

For the initial stages of the project it's decided to use ARIMA model to perform the time series analysis. This is been chosen initially with the reference of some of the tutorials i have seen and some of the blog posts. Different models has been used by the end of the project to achieve the end results.

ARIMA Model

Some of the important characteristics to consider when dealing with time series data are:

– Is there a Trend (measurements increase or decrease) over time?
– Is there a seasonality, meaning that there is a regular patterns in the data which are repeating over. This can be every year, month, day, week or season ?
– Is there a consistent variance over time?
– Are there any outlines and abrupt changes which are not related to seasonality?
– Is the data stationary?

The above mentioned characteristics are checked for the data using the Statsmodel framework which has all the time series analysis algorithms.

The seasonality and the trend of the data can be seen in the below figure.

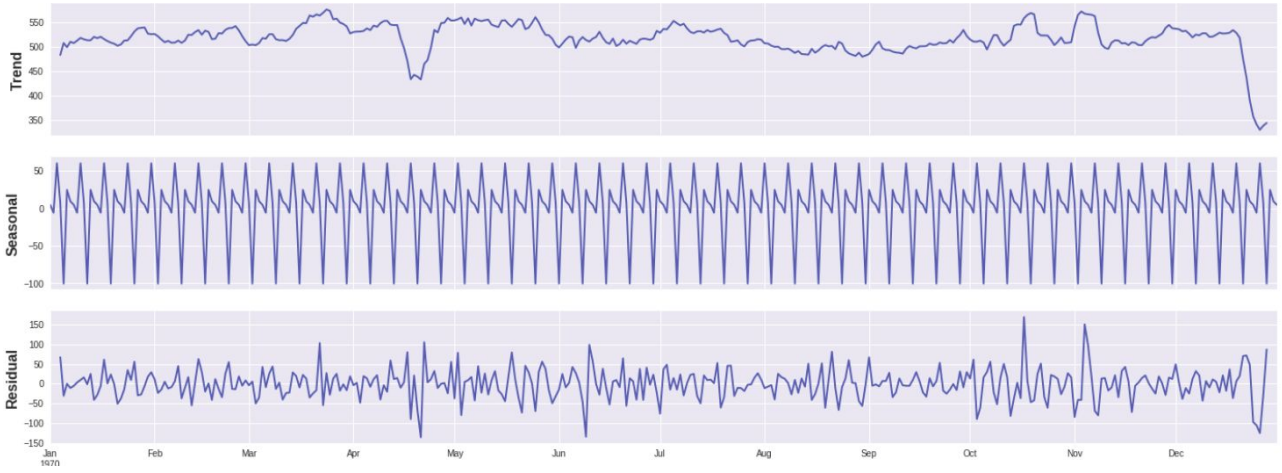

Fig4: seasonality and trend using statsmodel

We can see from the above figure there a clear monthly seasonality in the data. The seasonality in the data usually lead to non stationarity of the data. This can be confirmed by doing a Dickey-Fuller test on the data which will tell if the data is stationary or not.

```
 Augmented Dicky-Fuller Test
ADF Test Statistic : -1.2005248563537765
p-value : 0.6733698912369828
# of lags : 21
Num of Observations used : 1074
weak evidence against null hypothesis
Fail to rejct null hypothesis
Data has a unit root, it is non -stationary
```

Fig5: Dicky-Fuller Test results on the given data

The Augmented Dickey-Fuller test confirmed that the data is non- stationary. To apply the ARIMA the data needs to be stationary. So to achieve that, differencing technique had been used where the difference of the value at different lags has been calculated to see how the value is changing for each lag. Usually this can be started with First-order differencing, where the value is shifted by 1 lag. The results of the first order differencing were as followed.

```
adf_check(per_day['First Difference'].dropna())
```

```
 Augmented Dicky-Fuller Test
ADF Test Statistic : -10.067910457000123
p-value : 1.281656570421125e-17
# of lags : 20
Num of Observations used : 1074
Strong evidence against null hypothesis
reject null hypothesis
Data has no unit root and is stationary
```

Fig6: Dicky-Fuller Test results after first order difference

From the results of the test we can bee that the data is stationary now. This data can be applied to the ARIMA model. For the better accuracy of the model it was suggested that seasonal differencing would be better so the seasonal differencing has been performed on the first order differencing.

```
adf_check(per_day['Seasonal First Difference'].dropna())
```

```
 Augmented Dicky-Fuller Test
ADF Test Statistic : -8.71282722117961
p-value : 3.552970701924573e-14
# of lags : 21
Num of Observations used : 1067
Strong evidence against null hypothesis
reject null hypothesis
Data has no unit root and is stationary
```

Fig7: Dicky-Fuller Test results after seasonal first order difference

The result from the test confirmed that the data is still stationary. So this data was given to the ARIMA model. The ACF has been used on the data to see the relation between the lagged data.

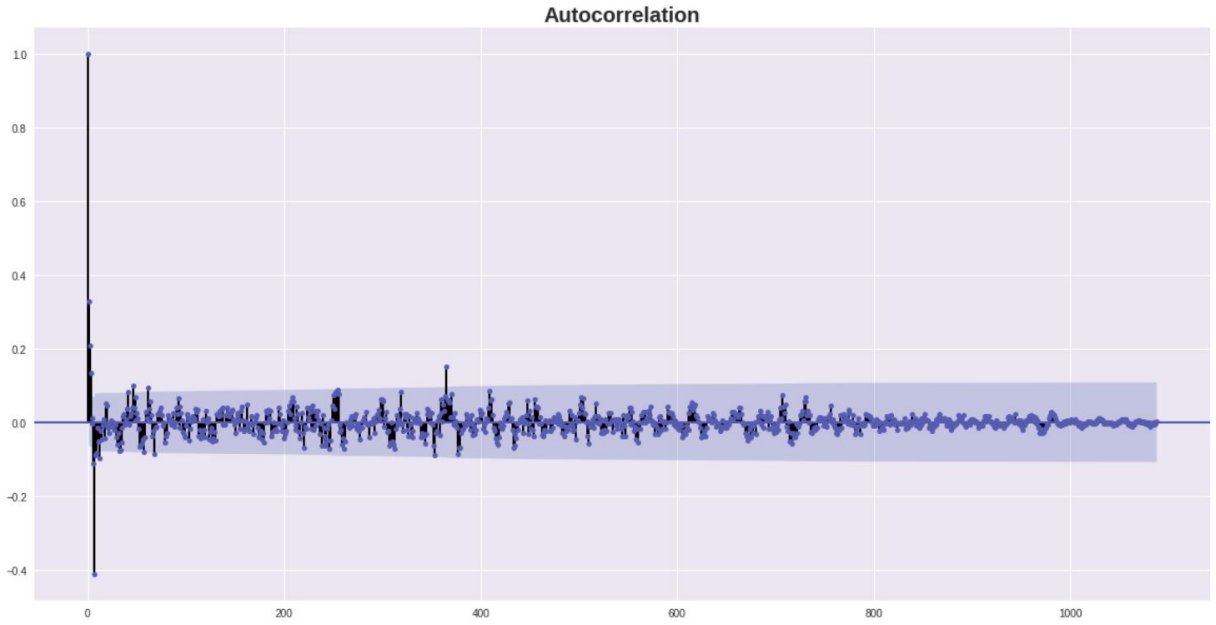

Fig8: ACF result

This shows that there is a strong relation between the actual values and the lagged values.

To apply the ARIMA model, the seasonal order must be identified to fit the values to the model. A script was written to identify the the seasonal order components from the data, but the script was taking too long to executed and there was no output from the script. So a default seasonal order of (1,2,1,12) has been chosen based on some references. The model fitted to the known data well but when predicting the next values the model failed to predict all the values in the month, instead it just predicted the average value for all the months.

Fig9: ARIMA prediction

The results from the ARIMA model were not promising and did not help answer the original question. This resulted in searching for another framework which can deal with time series data. After some research the framework from Facebook research team named Facebook Prophet has been chosen to find a suitable model for the data.

Prophet Model

Prophet which follows sklearn API, takes two inputs, one is a timestamp in datetime data type and other one is the numeric to predict, in our case the number of bookings per day.

The prophet model also allows to specify the holiday data which usually causes the seasonality in may business data. Prophet model let the user define specific holiday list so the model can know that this dates are special the will be treated differently while model processing. With the help of good documentation it did not take much to design the prophet model compared to that of the ARIMA model. The prophet model is mainly used on daily data which is what we have here so this model seemed most suitable for the task.

The prophet model identifies some change point in the data which are different from the holidays which are provided for the model. These change point are the dates where there is drastic change in the data even with being a holiday data. This change points can also be holiday dates in some cases. By knowing the change point we can see how they are affecting the model accuracy and determine if they are important to mention in the model, so that the model can specifically know that these are the change points which are making the most difference.

For the holiday data there is a separate holiday list for all the federal states of Germany which is included in to the Dataframe as a list so that the list can be passed to the model. The model also lets us see different trends like weekly, monthly, and daily.

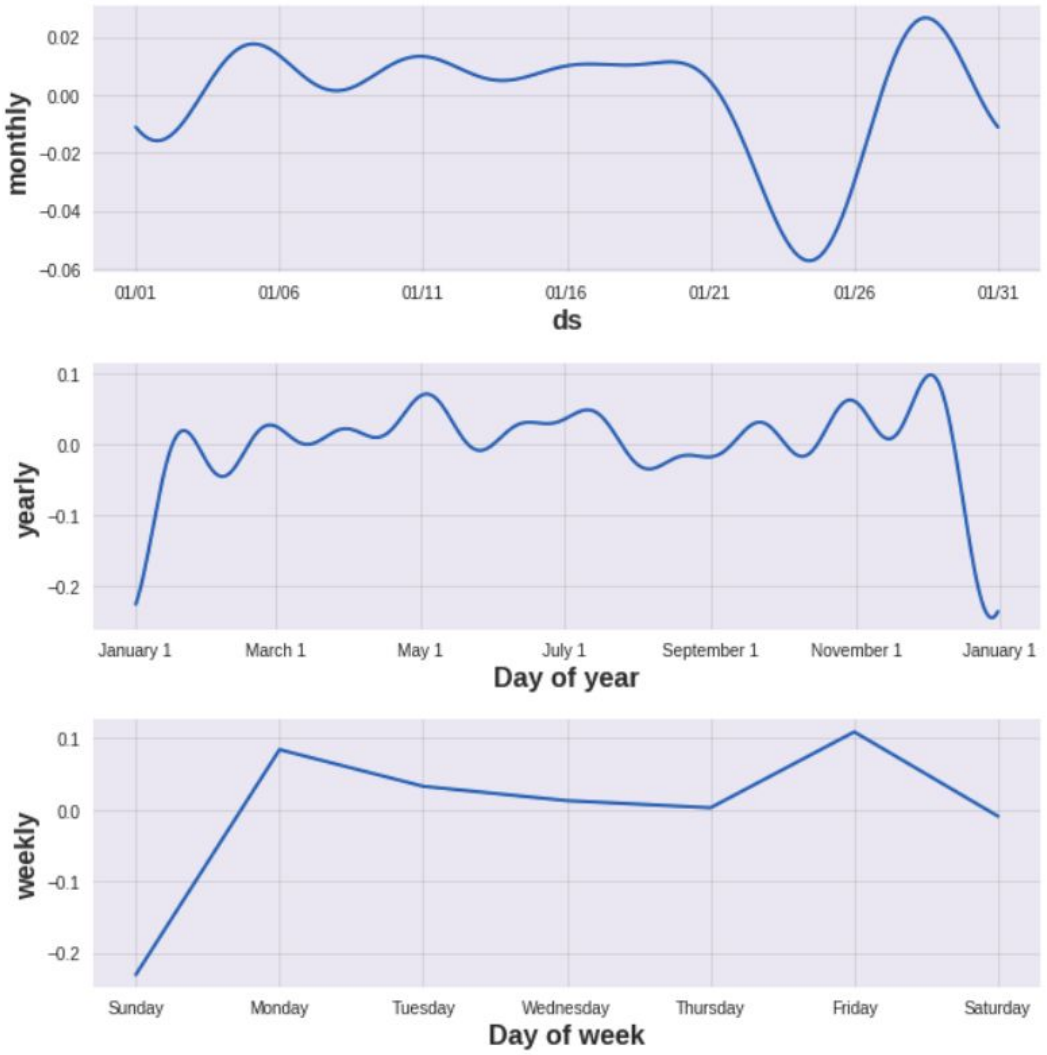

Fig10: Trends from the prophet model

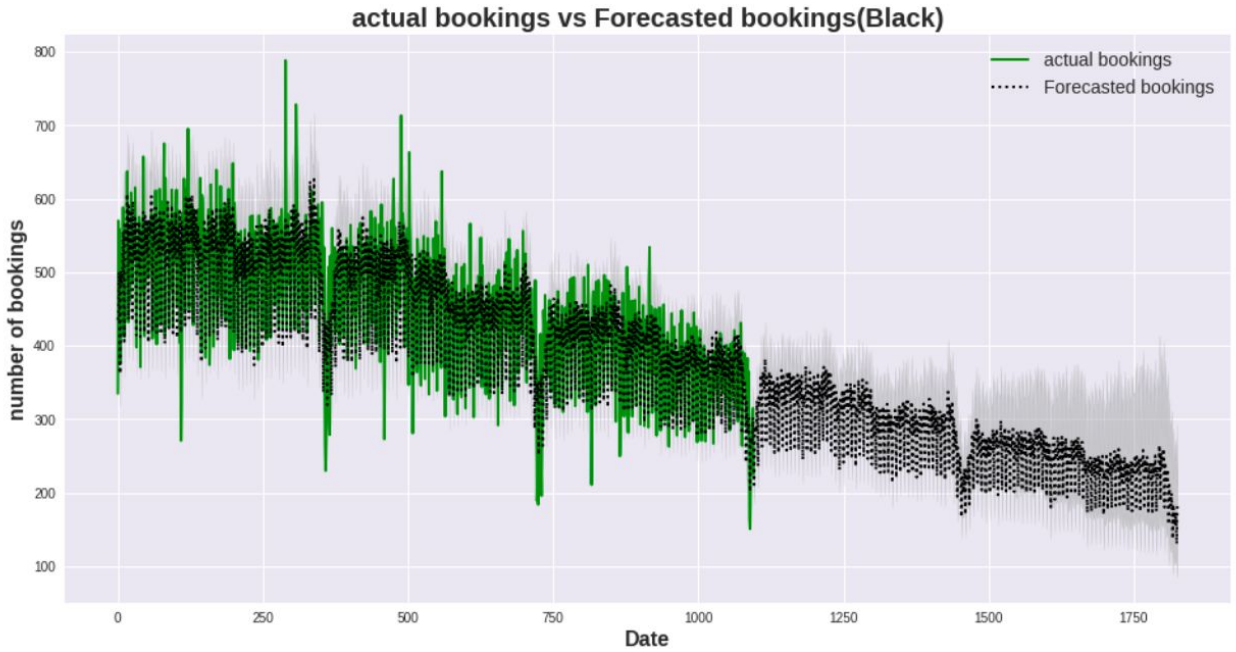

Fig11: forecast results from prophet model with out holiday data

As we can see that the data is fitting to the model with even following some of the out liners. The RMSE value for the data without knowing the holiday data was 42.78403370588128. The model was improved with introducing the holiday data. The model was first tested on the overall data i.e for all the cities. After checking accuracy, the model was used for each city and corresponding federal state holiday list.

The RMSE value decreased after the holiday data included into the model. 40.99875376421798 is the new value for the data.

Based on this, several models have been tried to find the appropriate model. The models were compared based on the RMSE values. Since I had some knowledge on neural networks several, I had build some of the neural networks with the help of some of the tutorials. The KERAS python library which has been developed on top of TensorFlow to make it easy for developing the models.

Effect of changepoint_prior_scale:

The changepoint_prior_scale is the trade-off between underfitting and overfitting of the model. If the value is high then the model will over-fit the data and if the value is less then the model will under-fit the given data. This can be related to the bias-variance trade-off in machine learning. To choose the appropriate changepoint_prior_scale value a script has been written which printed all the RMSE value of all the model which is used to measure the accuracy of the model. From the observed RMSE values the appropriate changepoint_prior_scale value has been chosen for the further models.

The value 0.4 was the best fitting changepoint_prior_scale value. The RMSE value with the use of this value is low. So this value has been chosen for training the model.

Below is the predicted values for the time period January 2017 till the June of 2017. the data we have is till June 2017, so now we can see how good out model is doing on completely unknown data which it has not seen.

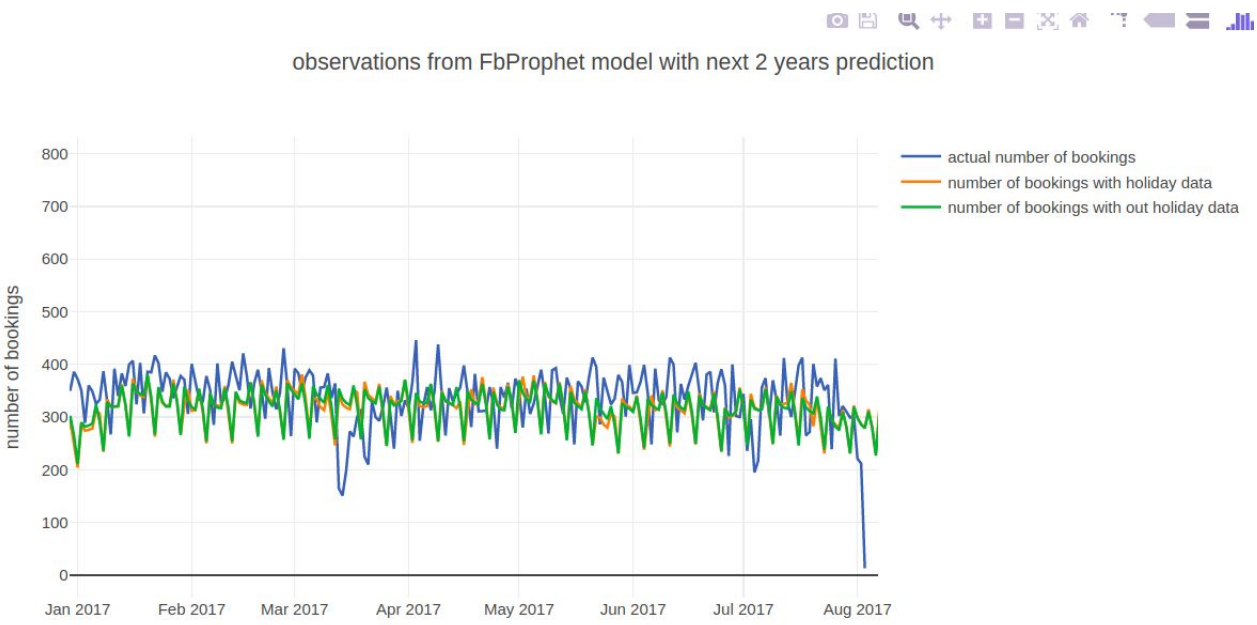

Fig12: comparing the predicted values with data that model has not seen.

Neural Networks

The time series data we have is more suitable for supervised learning, the previously observed number of bookings per day has been used to train the layers of the network. This can also be done with unsupervised learning using some of the generative deep learning models. For the simplicity of the project only some of the supervised learning algorithms were used.

Since this is the first time I have tried to use the neural networks in practice, i had taken some time to learn something and also to try different things out so that I could become familiar with more things. The models was tried with different ratios of the Train/Test compositions like 60/40, 80/20 and 90/10, where 60 means 60% of the total data has been used to train the model and 40% of the other data has been used to test the data on already trained model. The resulting RMSE values can be seen in Fig16 and Fig17.

Deep neural networks regression

The DNNregressor model from TensorFlow was tried to find if there can be any interesting results from the model but it resulted in fitting the data too close with a loss of 0.184, so this was considered in the evaluation process. The results from the model can be seen below.

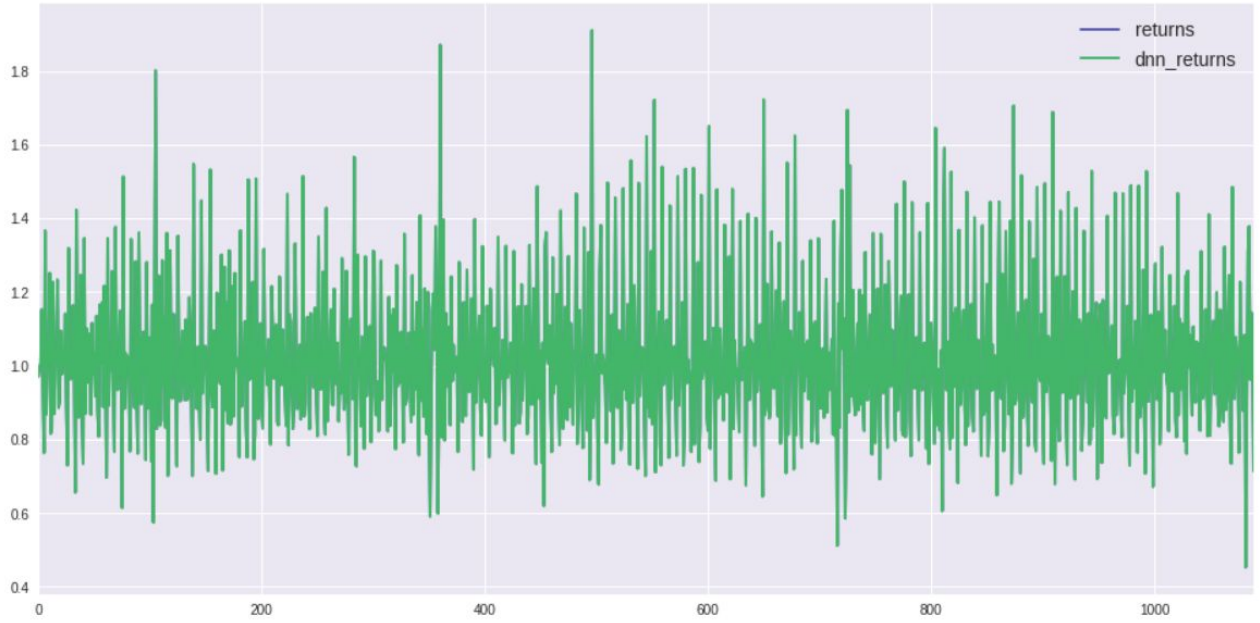

Fig13: DNNregressor predicted values.

The second model which has been used was a simple multilayer perceptron with a window operator by lagging the data.

Multilayer Perceptron(MLP)

The multilayer perceptron is a feed forward neural network with several layers in it. The input layer will get the inputs and the output of the input layer is given as input the hidden layer which is in between the input layer and output layer. The output of the hidden layer is given as input to the output layer. There can be multiple output layers in practical, since out data has only one output at the given point there is only one output needed from the output layer with one unit in it.

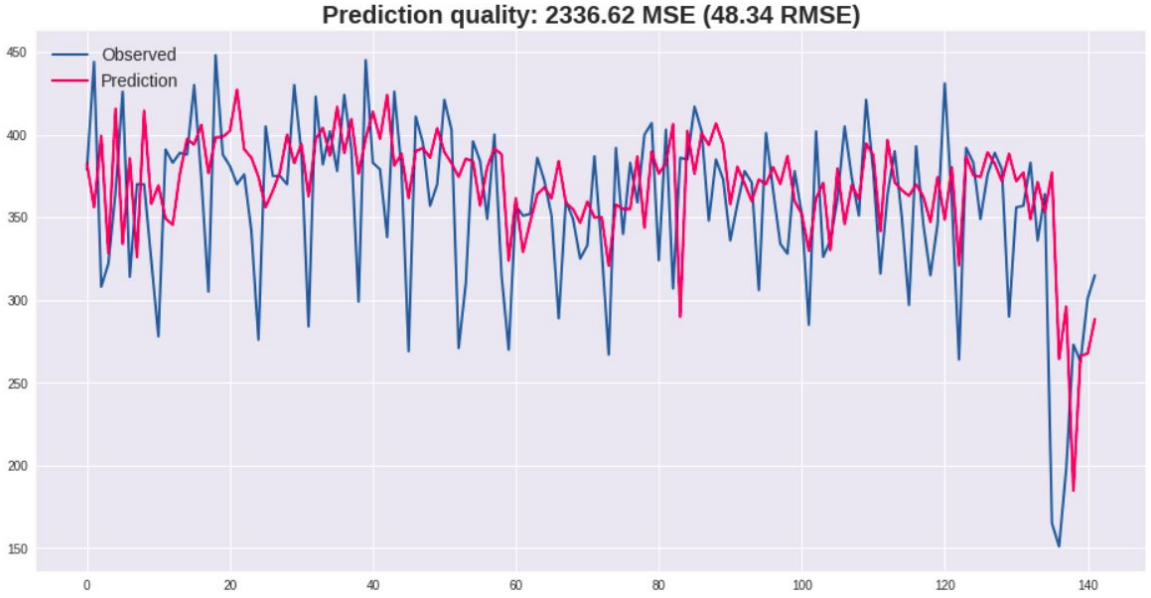

Fig14: test dataset output for MLP

As we can see from the above figure the model seem to be doing a reasonable job on the test data with a RMSE value of 48.34 the plot plot can be understood by lagging back the predicted values. The lag is used to differentiate the predicted values. The plot shows that it is picking up the pattern clearly but the model is not doing that great with the out liners or the sudden changes in the data, which is very hard to predict due to different contributing factors.

The MLP was chosen with 4 input nodes/neuron at the input layer and the hidden layer was designed with 12 neurons which are given to the single neuron at the output level. This network was chosen after testing several random combinations, as there is no definitive way to select the network complexity.

In neural networks when we are dealing a with a time series data, training recurrent network will give a better results than a feed forward network, because the time series data has some patterns which a recurrent network can remember with the memory it has. So the special kind of recurrent neural networks which are capable of learning short term dependencies know as LSTM(Long short term memory) had been used to train and test the data.

LSTM(Long short term memory)

The long short term memory(LSTM) is one of the special kinds of recurrent neural networks which remembers certain patterns for some time, in this case for certain number of iterations and use this patterns to predicted output for the newly given values. This type of neural networks are of a great use when the data had a historical property with repeating patterns over time. This allows the recurrent nets to learn throughout the time periods, so that they can link cause and effects in the data.

In this model the input layer is with 3 neurons and the hidden layer a LSTM layer with 12 neurons in it. This was also chosen based on the RMSE values for the corresponding values. The output layer contains one neuron with the output being a single value. For the LSTMs, the activation function ‘softsign’ has been used over ‘relu’ because this is more faster and less prone to saturation.

The model accuracy was decreasing when the epochs of the network increased. This can be due to the fact that the data set is small and there are more number of iterations happening with the small amount of the data. This leads to changing the weights in the activation function many times which results in decreasing accuracy.

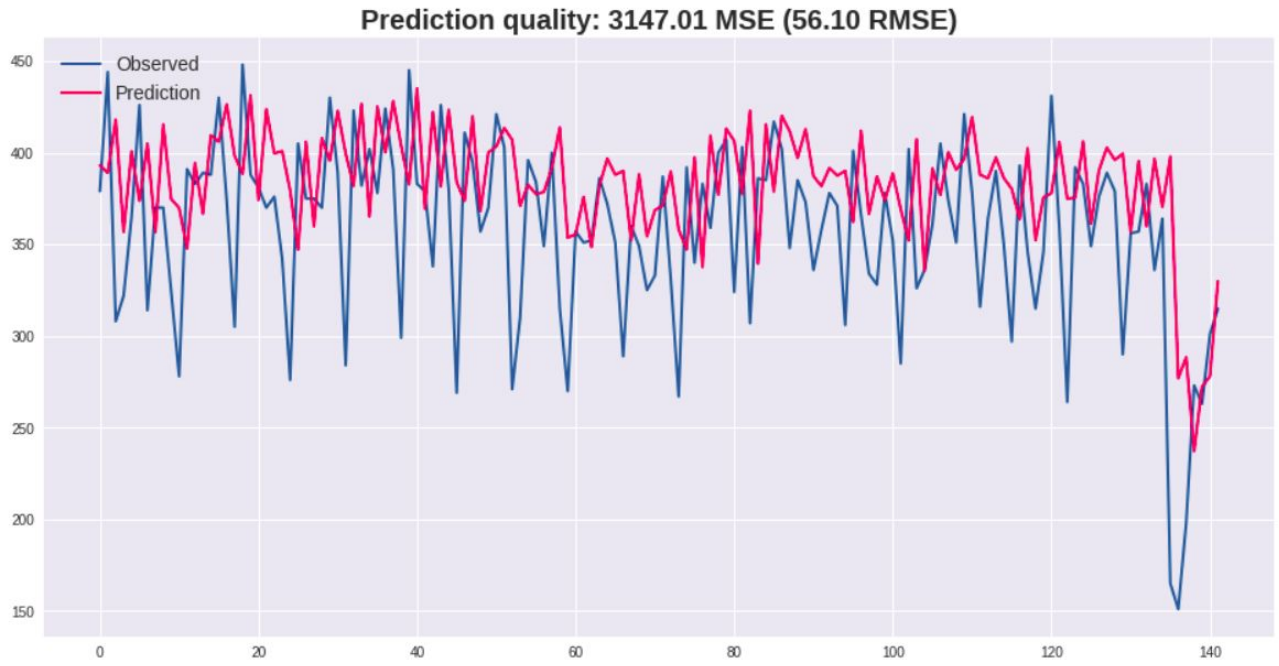

Fig15: LSTM predicted values

The results from LSTM are a bit also were promising but the model is failing to detect the usual peaks and lows of the data, this can be maybe because of the fact that we have very less number of observations. The LSTM usually does well when we have a large number of data, so this can might be one of the reasons for the less accurate model accuracy which resulted in higher RMSE value.

Train/Test(60/40)

Multilayer perceptron with window

Train Score: 4912.00 MSE (70.09 RMSE)

Test Score: 3156.05 MSE (56.18 RMSE)

LSTM Recurrent Neural Network

Train Score: 68.37 RMSE

Test Score: 104.67 RMSE

Train/Test(80/20)

Multilayer perceptron with window

Train Score: 5025.14 MSE (70.89 RMSE)

Test Score: 3063.79 MSE (55.35 RMSE)

LSTM Recurrent Neural Network

Train Score: 68.65 RMSE

Test Score: 61.11 RMSE

Fig16: RMSE values for 60/40, 80/40

Train/Test(90/10)

Multilayer perceptron with window

Train Score: 4309.63 MSE (65.65 RMSE)

Test Score: 2336.62 MSE (48.34 RMSE)

LSTM Recurrent Neural Network, epochs=100

Train Score: 67.02 RMSE

Test Score: 56.10 RMSE

LSTM Recurrent Neural Network, epochs=200

Train Score: 67.20 RMSE

Test Score: 60.13 RMSE

Fig17: RMSE values for 90/10

From the above observed values we can see that prophet model results in less RMSE value for the data, so the prophet model has been used for the remaining data analysis.

The analysis which has been performed until now was based on the number of booking per each day. The number of vehicles used on each day can be predicted with the model we have built. The results form the prediction can be seen below.

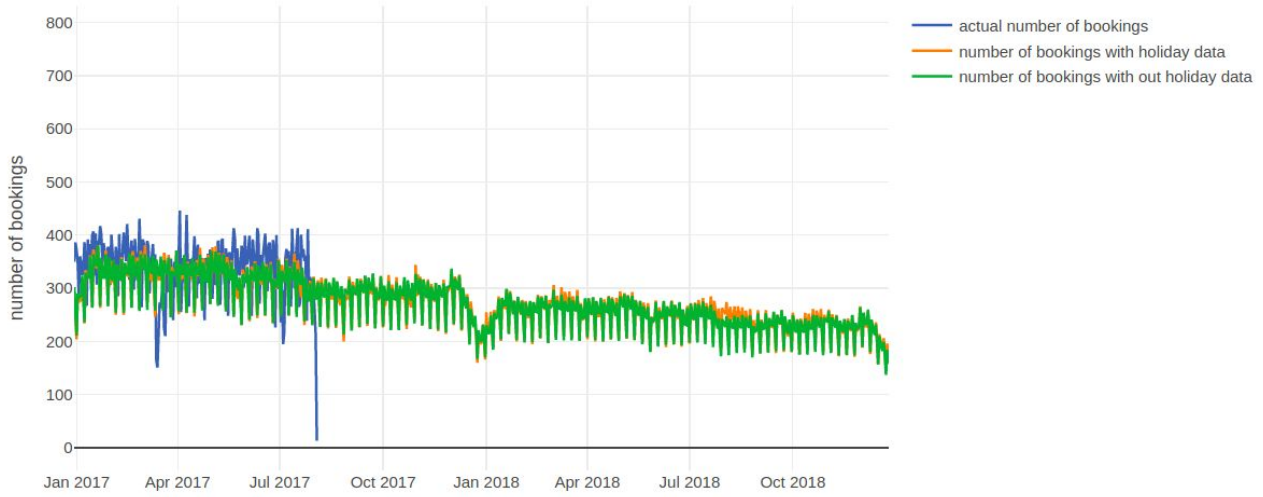

Fig18: Number of bookings per day.

observations from FbProphet model with next 2 years prediction

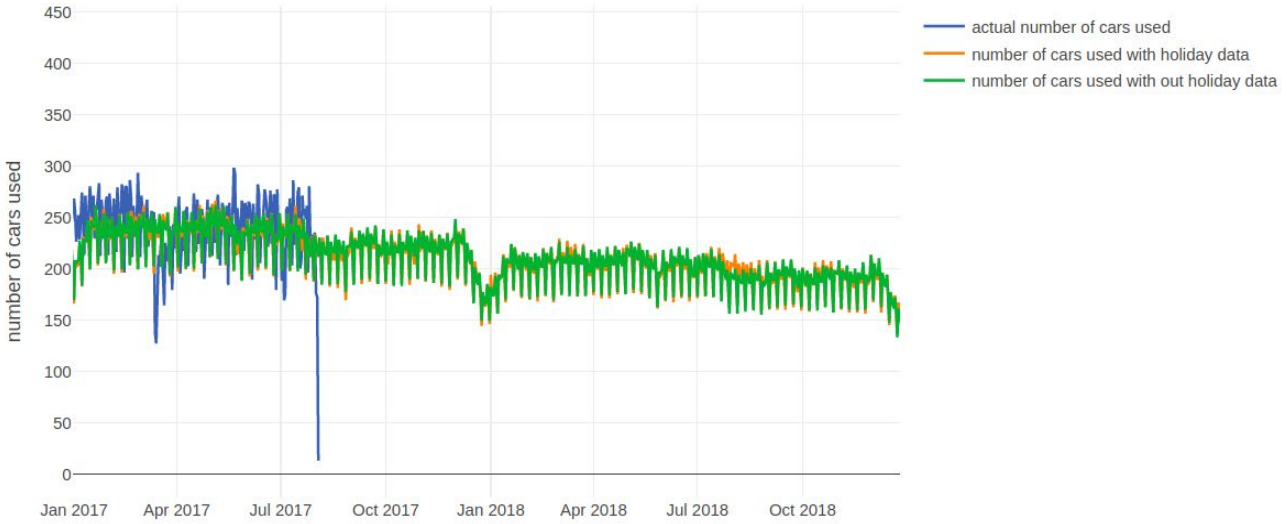

Fig19: Number of cars needed per day.

The effect of the holidays is minor on non holiday days, as it can be seen in the plots. The number of bookings, number of cars used for each city in a day can be seen with a little changes in the code. Thus the two solution for the both questions framed in the beginning of the project can be answered with the results shown above.

GitHub project link: https://github.com/jawaharreddy/Db-Data-Analysis

Conclusion

In this documentation, the task of analyzing the bookings data from car sharing company Flinkster was performed. The project started with two students, but unfortunately only one student was able to be there until the end of the project. The data analysis was performed based the business questions which were formed at the beginning of the project. The analysis was supposed to answer this questions with help of some other data combined with the booking data. An initial data analysis was performed to know more about the data which helped in forming the questions. Time series analysis was performed on the data to get the results. different models were built using different libraries available in python. The prediction of number of bookings happening on a particular day and number of cars needed for a particular day for each city and on the overall data using Prophet model which was chosen by comparing the accuracy of the model with different models. The final results were documented accordingly and all the referred materials are cited in references section.

References

1) “Getting started with Prophet” https://facebook.github.io/prophet/docs/

2) “Sequential model guidance” https://keras.io/getting-started/sequential-model-guide/

3) “A beginner's guide to Recurrent networks and LSTM” https://deeplearning4j.org/lstm.html

4) “Understanding LSTM” https://colah.github.io/posts/2015-08-Understanding-LSTMs/

5) “How to remove seasonality and trends” https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python

6) “Time series analysis in python” https://towardsdatascience.com/time-series-analysis-in-python-an-introduction-70d5a5b1d52a

7) “Important steps in data analysis” https://www.bigskyassociates.com/blog/bid/356764/5-Most-Important-Methods-For-Statistical-Data-Analysis

8) “Time series analysis with neural nets in python” http://dacatay.com/data-science/part-6-time-series-prediction-neural-networks-python/