New York City Taxi Fare Prediction

Capstone Project

Machine Learning Engineer Nanodegree

By

Gagan Saini

April 14, 2019

Contents

CHAPTER 1

DEFINITION

1.1 PROJECT OVERVIEW

With the surge in app-based taxi providers, it has become very important to predict the taxi fare for a trip in advance. The user can then decide whether he wants to avail the ride or not. Also, as there are number of providers, a user can compare the fare of different providers for a trip and then choose the best one.

One simple way to estimate the fare is by using total distance of the ride and then calculate the fare based on the number of miles covered and price per mile. But the price prediction in real world is much more complex than that. Various other factors may affect the pricing like toll taxes, number of passengers, pickup and drop-off points, time of the day the ride is taken, advance booking etc. Some other variable factors are also considered by the taxi providers these days. For example, the prices will be high in case of high demand which is usually called surge pricing, the route taken by for the ride etc.

Fare prediction in advance directly affects the business because that is the most important factor based on which the customer will decide whether he wants to go with the taxi provider or not. So, it is very important to predict the fare as much accurately as possible.

There are many academic research papers trying to solve the problem of fare & time prediction. The time taken can also directly affect the total fare of a ride. In [1], they have applied Support Vector Regression (SVR) to predict the time taken for a ride in advance. In [2], Neural Networks (SSNN) have been used for time prediction. In [3], they have used deep neural networks for the fare prediction.

1.2 PROBLEM STATEMENT

To predict the fare amount for a taxi ride in New York City given the longitude and latitude coordinates of the pickup and drop-off locations, date & time of the pickup and number of passengers.

This is clearly a regression problem. Regression is used to predict the value of a dependent variable based on a number of independent variables.

In this problem, the fare to be predicted is the dependent variable and the features like pickup and drop off locations, number of passengers, and date time of pickup are the dependent variables. We may deduce new feature out of the given features like distance traveled, time of the ride, weekday etc. More details about the data are presented in section 2.1.

The problem can be solved by using regression techniques. I will try a handful of regression techniques and will choose the one which works best in terms of predicting the fare. I will use Linear Regression, Decision tree regressor, SVM and Ensemble methods.

1.3 METRICS

RMSE (Root mean Squared Error) is used as an evaluation metrics in the original Kaggle Competition[4]. So, I will also use RMSE for the evaluation of my models.

RMSE measures the difference between the predictions of a model, and the corresponding ground truth[5]. It is a measure of the spread of actual y values against the predicted y values. RMSE is given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1.1}$$

Where, y_i is the truth value and $\hat{y}_i$ is the predicted value.

A large RMSE means a large average error. So smaller values of RMSE are desirable.

CHAPTER 2

ANALYSIS

In this chapter, I will discuss about the data set, algorithms used and the benchmark model.

2.1 DATA EXPLORATION

The dataset is provided by Kaggle [4]. It contains the following three files:

2.1.1 Data Set

1. train.csv - Input features and target fare_amount values for the training set (about 55M rows).

2. test.csv - Input features for the test set (about 10K rows).

3. sample_submission.csv - a sample submission file in the correct format (columns key and fare_amount). This file 'predicts' fare_amount to be $11.35 for all rows, which is the mean fare_amount from the training set.

As the test.csv file does not have the corresponding value of the target variable. We can not use this data to verify our model. So, I will use the data from the train.csv file only by splitting the data into train, validation and test sets.

Features

The data set contains the following features:

Table 2.1: Data Features

Column Name	Data Type	Description
pickup_datetime	timestamp	The date and time when the ride started.
pickup_longitude	float	Longitude coordinate of where the taxi ride started.
pickup_latitude	float	Latitude coordinate of where the taxi ride started.
dropoff_longitude	float	Longitude coordinate of where the taxi ride ended.
dropoff_latitude	float	Latitude coordinate of where the taxi ride ended.
passenger_count	integer	Number of passengers in the taxi ride.

Target Variable

The data set contains the following target variable :

Table 2.2: Target Variable

Column Name	Data Type	Data Range	Description
fare_amount	float	-$300 to $93963.36	Total cost of the taxi ride in dollars.

The data set contains one more column named 'key'. This is not useful for the model training. This column is only used for the identification of rows, therefore will be removed before model training.

Some example rows from the data set can be seen in Fig 2.1

	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2009-06-15 17:26:21.0000001	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	2010-01-05 16:52:16.0000002	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	2011-08-18 00:35:00.00000049	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2
3	2012-04-21 04:30:42.0000001	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1
4	2010-03-09 07:51:00.000000135	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1

Figure 2.1: Data Example

Basic statistics about the data are presented in Fig 2.2

The statistics shown above can help us in finding out some basic characteristics of the data. I found the following abnormalities in the data which need to be taken care of before proceeding with the model training:

- The fare_amount is varying between -1.000000e+02 to 1.273310e+03. The minimum fare amount is negative here, which is not possible until some

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	5.000000e+06	5.000000e+06	5.000000e+06	4.999964e+06	4.999964e+06	5.000000e+06
mean	1.134080e+01	-7.250678e+01	3.991974e+01	-7.250652e+01	3.991725e+01	1.684695e+00
std	9.820175e+00	1.280970e+01	8.963509e+00	1.284777e+01	9.486767e+00	1.331854e+00
min	-1.000000e+02	-3.426609e+03	-3.488080e+03	-3.412653e+03	-3.488080e+03	0.000000e+00
25%	6.000000e+00	-7.399206e+01	4.073491e+01	-7.399139e+01	4.073404e+01	1.000000e+00
50%	8.500000e+00	-7.398181e+01	4.075263e+01	-7.398016e+01	4.075315e+01	1.000000e+00
75%	1.250000e+01	-7.396711e+01	4.076712e+01	-7.396367e+01	4.076811e+01	2.000000e+00
max	1.273310e+03	3.439426e+03	3.310364e+03	3.457622e+03	3.345917e+03	2.080000e+02

Figure 2.2: Data Statistics

cashback or refund is there. But in our problem we don't consider any such cases, so we'll remove the rows with negative or zero fare.

- The minimum value of passenger_count is zero here, which is not possible. There should be at least one passenger taking the ride.
- Similarly the maximum value of the passenger_count as 208 is very high.
- The longitude and latitude coordinates are way beyond the allowable range of these coordinates. Which is (-90, 90) for latitude and (-180, 180) for longitude.
- Also, as we are considering the taxi fare prediction in the Newyork city only. We need to restrict the longitude and latitude coordinates within the coordinate range of the city.

2.2 EXPLORATORY VISUALIZATION

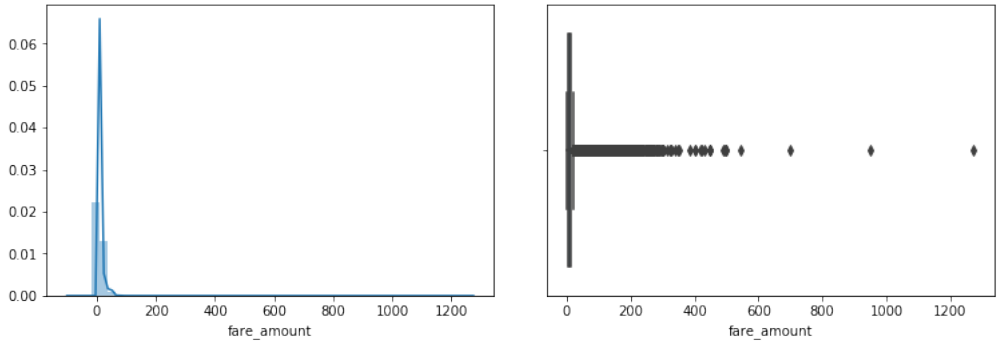
Lets try to find out the patterns and other insights into the data using various diagrams.

We can see the distribution of fare amount across various rides, in Fig 2.3a and fare amount box plot in Fig 2.3b. We can see that the data contains many outliers.

Next, we will see the histogram of passenger counts in the data in Fig 2.4. We can see that there are some entries in which the passenger count is zero and in some rows this count is way beyond the reason.

Now, lets see the scatter plots of pickup and dropoff coordinates in Fig 2.5. The range of latitude and longitude coordinates is way beyond the allowable range.

We can draw some other graphs w.r.t. the pickup_datetime but that data is currently in string format. Later we will convert this into date format and then split the date into various components like month, year, day of week etc in section 3.1.



(a) (b)

Figure 2.3: Fare Amount Distribution
(a) Distribution plot (b) Box plot

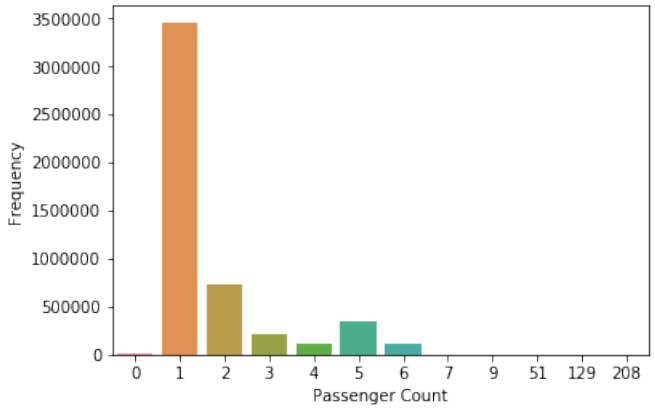


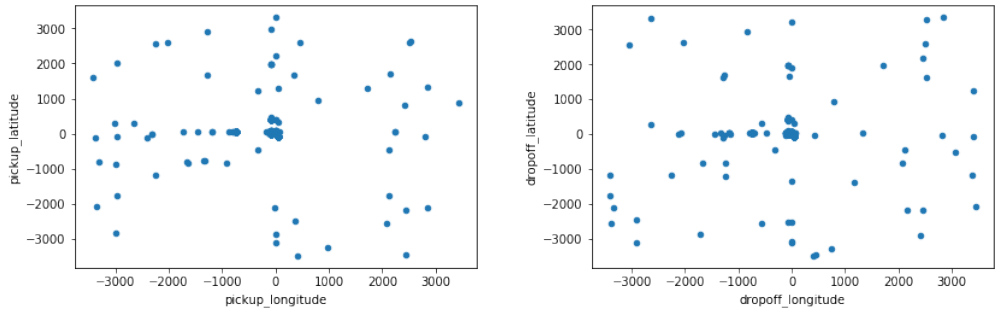
Figure 2.4: Passenger Count Histogram

2.3 ALGORITHMS AND TECHNIQUES

Taxi fare prediction is clearly a type of regression problem. Regression techniques are used to predict the value of a dependent variable based on a number of independent variables. In this problem, the fare to be predicted is the dependent variable and the features like pickup and drop off locations, number of passengers, and date-time of pickup are the dependent variables. We may want to deduce new feature out of the given features like distance traveled, day of week etc. I will use the following regression techniques to solve this problem:

2.3.1 Decision Trees

Decision trees[6] can be used for both classification as well as regression problems. It predicts the value of target variable by inferring some simple rules from the features present in the input data. It does so by down the data into smaller subsets based on some rules inferred from input features and incrementally creates a tree. The leaf node



(a) (b)

Figure 2.5: Scatter Plot of Location Coordinates
(a) Pickup Longitude vs Pickup Latitude (b) Dropoff Longitude vs Dropoff Latitude

represent the final value of the prediction.

Advantages

- Simple to understand and visualize.
- Requires little data preparation
- Able to handle both numerical and categorical data.
- The cost of predicting the data is logarithmic in number of data points used to train the tree.

Disadvantages

- Can result in complex trees leading to overfitting.
- Decision trees can be unstable as even a small change in data can lead to a significantly different tree.
- If some classes dominate, decision trees learners can create biased trees.

2.3.2 Random Forests

A Random Forest[7] estimator fits a number of decision trees on different subsets of data. Whenever a new data point comes, it runs it through all the decision trees, and then find out the estimate by averaging the output from different decision trees as shown in Fig2.6.

Advantages

- It do not suffer from overfitting.

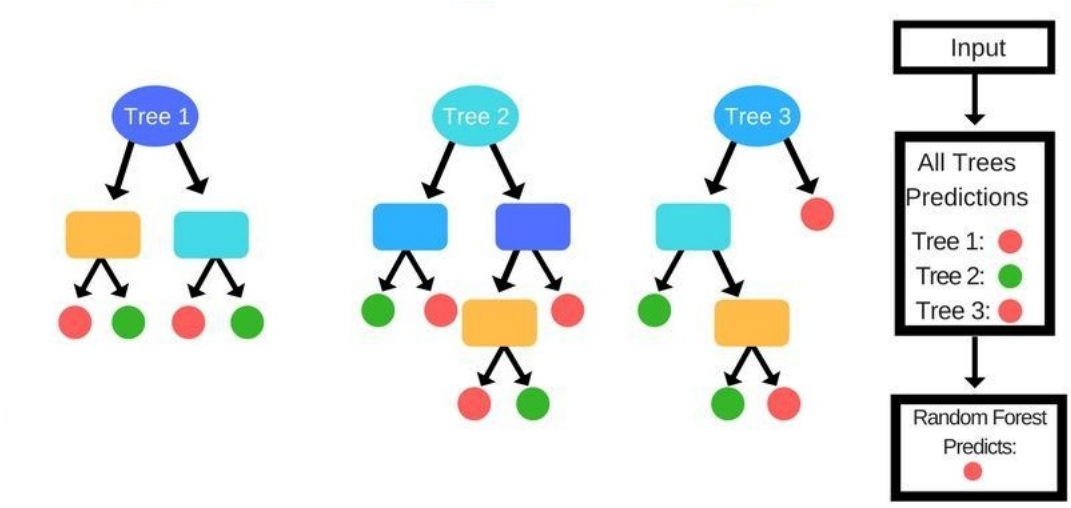


Figure 2.6: Random Forest Estimator[8]

2.3.3 Support Vector Machines

Support Vector Machines[9] are a set of supervised learning methods that can be used for both classification and regression. Given the training data, SVM outputs an hyperplane which divides the data into different categories. We map the data into higher dimensions (kernel trick), where we can nicely separate the data as shown in Fig 2.7. Then we project the hyperplane in two dimensions to get the boundary that separates the data. Support vector regression works on the same principle. The model produced by Support Vector Regression depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

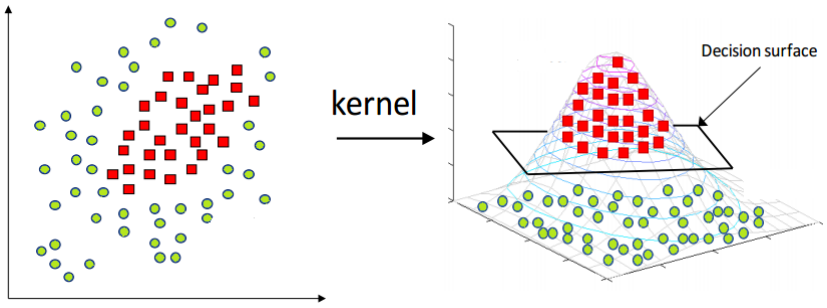


Figure 2.7: Support Vector Machine Kernel[10]

Advantages

- Effective in high dimensional space.

- Uses a subset of training points in decision function (called support vectors), so it is also memory efficient.

Disadvantages

- Can take large amount of time to train for large data sets.
- Parameter tuning e.g. choosing the right kernel is not straightforward.
- Difficult to understand and interpret the final model.

2.3.4 AdaBoost Regressor

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data[11]. AdaBoost is a meta-estimator which first begins by fitting a model on the original dataset. Then it assign higher weights to the points which are incorrectly classified and tries to classify these points correctly with a new model. This process is repeated for a fixed number of times. Finally the different models hence obtained are combined together based on the weights determined by the number of incorrectly and correctly classified points. This process is shown in Fig2.8 AdaBoost can be used for both classification as well as regression.

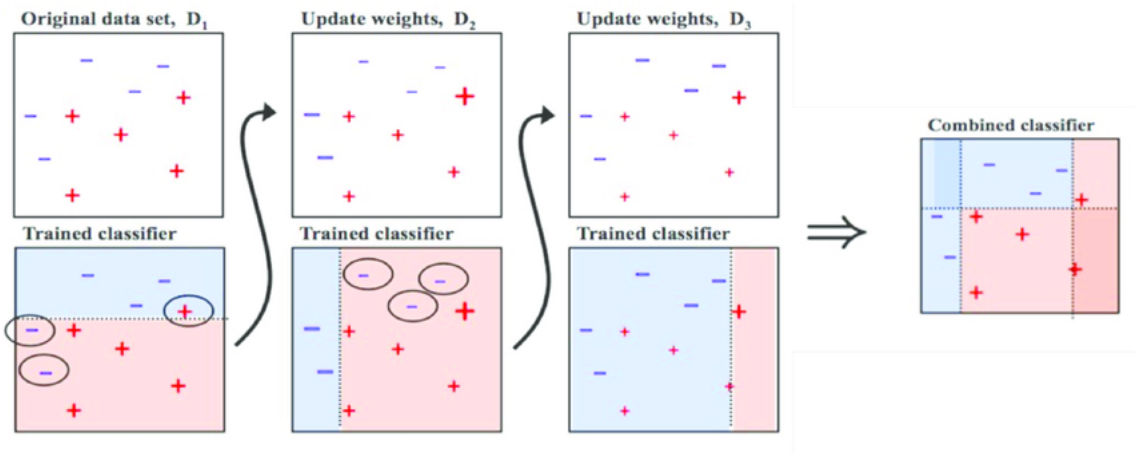


Figure 2.8: AdaBoost[12]

Advantages

- Simple. Combines several simple models.
- Generalizes well.
- Not prone to overfitting.

Disadvantages

- Sensitive to noisy data and outliers.

2.4 BENCHMARK

For benchmark model, I have used a simple linear model. A linear regression models tries to find out the linear relationship between the dependent and independent variables. A simple linear model is useful for finding relationship between two continuous variables.

One such model[13] is provided on the Kaggle in the problem description of this completion. I will compare the performance of other models with this model. The RMSE value of this model is given as to be 5.74184 on kaggle public score.

CHAPTER 3

METHODOLOGY

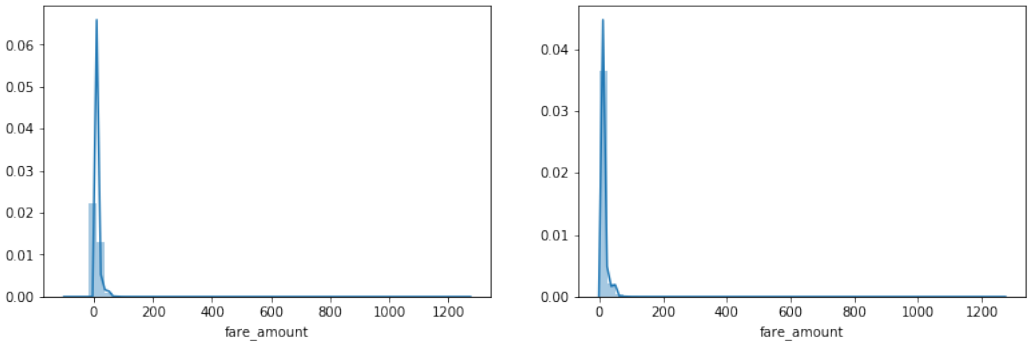
3.1 DATA PREPROCESSING

As we saw in section2.1, the data has many abnormalities. So In this section, we will try to cleanup the data by removing the abnormalities and outliers. We will clean the data by considering each column one by one.

As the size of the dataset is very large (55M rows), So it will require a huge amount of memory and processing power. I considered only first 50L rows for this project.

3.1.1 Fare Amount

Firstly, I Removed the rows with zero or negative fare amount.

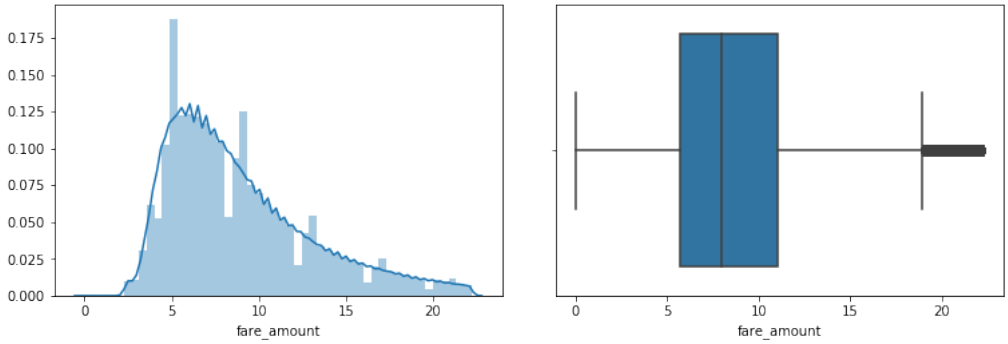


(a) (b)

Figure 3.1: Fare Amount Distribution
(a) Original Data (b) After removing non positive fares

I see the maximum fare amount goes beyond $ 1200. The data might have some outliers as it is not common or realistic to have taxi fares in this range. I tried to remove the outliers using Interquertile Range method which I learned in Udacity Course ”Intro to Descriptive Statistics”. The distribution curve and box plot I got after applying this method are quite satisfactory as shown in figure 3.2

But this process removed around 4,30,420 rows, which is a lot of data, So I decided



(a) (b)

Figure 3.2: Fare Amount Distribution after applying IQR method for outlier removal (a) Distribution curve (b) Box Plot

not to use this method to remove the outliers. And I removed the rows where fare amount is more than $200. The resultant distribution can be see in fig. 3.3.

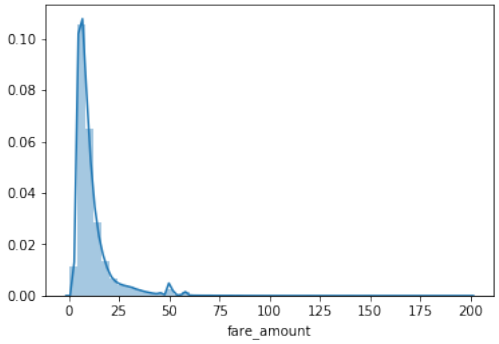


Figure 3.3: Fare Amount Distribution after removing fares > 200

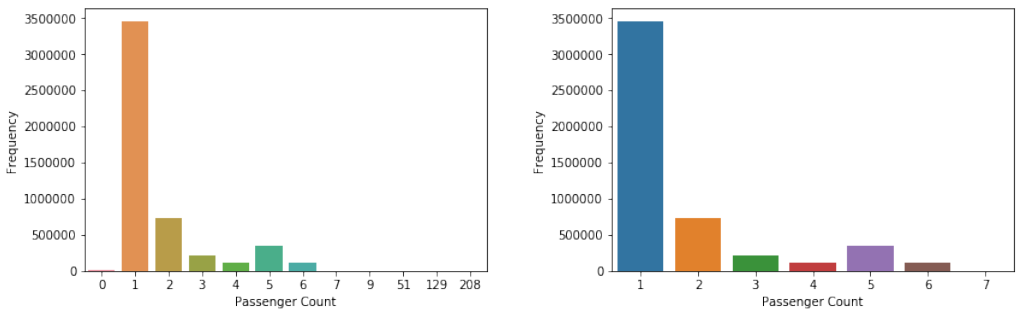
3.1.2 Passenger Count

There are a lot of outlier in passenger_count column where the passenger count is zero or very high around 200. So I removed the rows with passenger_count ≤ 0 and the ones with passenger_count > 7. The histograms before and after this step can be seen in fig. 3.4.

3.1.3 Longitude and Latitude Coordinates

As we saw in Data Exploration section 2.1, the longitude and latitude coordinates are well beyond the allowed range. So I applied the following steps to remove the outliers in these columns.

- Remove the pickup_latitude coordinates which are outside the range of [-90, 90]
- Remove the dropoff_latitude coordinates which are outside the range of [-90, 90]

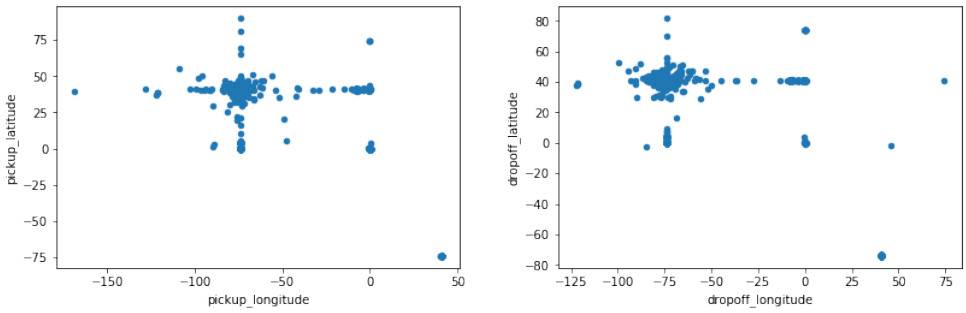


(a) (b)

Figure 3.4: Passenger Count Histogram
(a) Original Data (b) After removing passenger count outside (0,7]

- Remove the pickup_longitude coordinates which are outside the range of [-180, 180]
- Remove the dropoff_longitude coordinates which are outside the range of [-180, 180]
- Remove the rows where pickup and dropoff locations are same

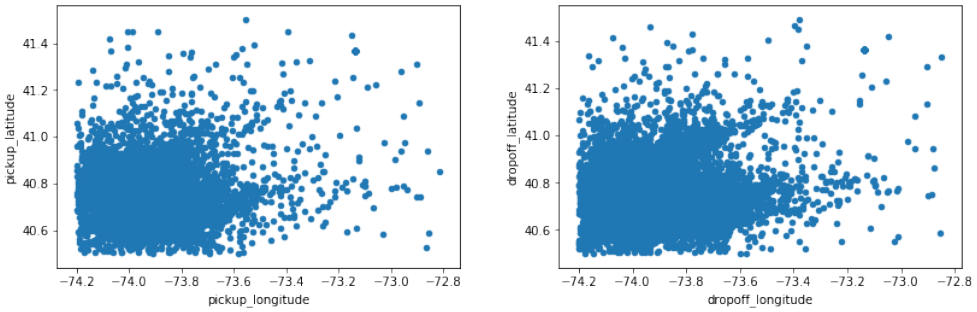
The scatter plots I got after applying the above steps are shown in fig. 3.5



(a) (b)

Figure 3.5: Location Coordinates Scatter Graph
(a) Pickup Coordinates (b) Dropoff Coordinates

Still there are so many outliers. As we know that the data consists of the rides within the Newyork city, so we can remove any coordinates that are outside the bounds of city. With the help of Google maps and latlong.net, I found the rough estimate of min and max coordinates of new york city with the longitudes ranging in [-74.2, -72.8] and latitudes ranging in [40.5, 41.5]. The scatter graphs obtained after removing the points outside the range of newyork city are shown in fig. 3.6.



(a) (b)

Figure 3.6: Location Coordinates Scatter Graph Within NYC
(a) Pickup Coordinates (b) Dropoff Coordinates

3.1.4 Duplicate or Missing Data

The data set neither had any duplicate rows nor the rows with any piece of data missing.

3.1.5 Feature Engineering

Feature engineering is a process of creating new features using the existing features present in the data. I created the following new features:

1. **trip_distance** in miles calculated by Haversine formula[14] using the longitude and latitude coordinates.
2. **pickup_year** from the pickup_datetime.
3. **pickup_month** from the pickup_datetime.
4. **pickup_day** from the pickup_datetime.
5. **pickup_hour** from the pickup_datetime.
6. **pickup_day_of_week** from the pickup_datetime.

Then I plotted some bar plots based on these new features to find out if they have some effect on the fare amount. Fig. 3.7a & 3.7b suggests that average fare amount is increasing every year. I see that number of rides are less in 2015 that is maybe because I have not considered the full data-set due to resource constraints.

Fig. 3.7c & 3.7d suggests that the avg fare amount is more for the hours when number of rides is less. And it is less when then number of rides is more.

Fig. 3.7e & 3.7f suggests that the trip count is higher on Friday and the fare is highest on Sunday when there are least number of trips.

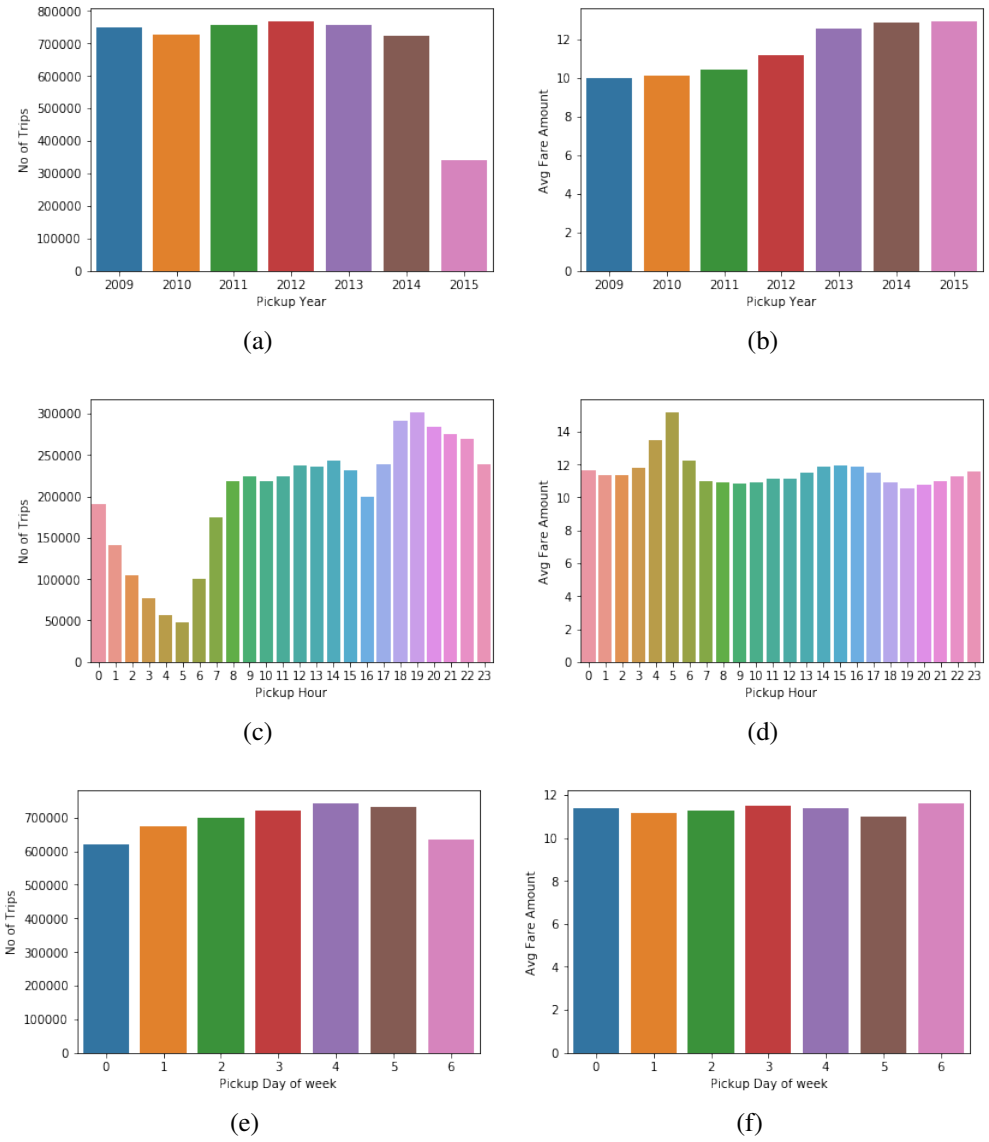


Figure 3.7: Affects of Pickup Year, Hour & Weekday on No of Trips and Fare Amount

3.1.6 Finalizing the Features

At last, I removed the unusable and redundant features which are:

1. **key** - Unique identifier for each row. Not helpful for predicting the fare amount
2. **pickup_datetime** - We have engineered new features from this column. This is redundant and will not be used now.

So the final features are shown in Fig. 3.8.

```
df.columns
```

```
Index(['fare_amount', 'pickup_longitude', 'pickup_latitude',
       'dropoff_longitude', 'dropoff_latitude', 'passenger_count',
       'trip_distance', 'pickup_day', 'pickup_hour', 'pickup_month',
       'pickup_year', 'pickup_day_of_week'],
      dtype='object')
```

Figure 3.8: Final Features

3.2 IMPLEMENTATION

As discussed in the section 2.3, I will apply various regression techniques and then choose the one which gives least RMSE value. Following are the common steps I followed for each of the algorithm:

1. Read the data set generated after the clean up done in section 3.1.
2. Split the data into training and testing sets by using train_test_split with a test_size of 0.33.
3. Fit the model with default parameters.
4. Calculate RMSE by making prediction using this model on test data.
5. Tune the hyperparameters by using GridSearchCV with 3 fold cross-validation.
6. Predict the fare_amount using the best model obtained after GridSearchCV on test data. Calculate the RMSE.

3.2.1 Decision Tree

I used DecisionTreeRegressor from sklearn.tree to implement this. By using the default parameters, It gave an RMSE value of 4.74 which is better than the benchmark model. It means the data cleanup and feature engineering has helped improving the prediction. I will further try to refine the parameters by using GridSearchCV from sklearn.model_selection.

Refinement

I ran GridSearchCV with the following parameter values:

```
parameters = {'max_depth':[20,100,150], 'min_samples_leaf':
[50, 100, 150], 'min_samples_split' : [50, 100, 200]}
```

It returned the following estimator as the best one:

```
DecisionTreeRegressor(criterion='mse', max_depth=20, max_features=None,
        max_leaf_nodes=None, min_impurity_decrease=0.0,
```

```
min_impurity_split=None, min_samples_leaf=20,
min_samples_split=100, min_weight_fraction_leaf=0.0,
presort=False, random_state=42, splitter='best')
```

The above model gave an RMSE value of 3.49, which is significant improvement over the model with default parameters. I tried to further tune the parameters, but couldn't get better performance than this.

3.2.2 Random Forests

I used RandomForestRegressor from sklearn.ensemble to implement this. By using the default parameters, and n_estimators as 100, I got an RMSE value of 3.31 which is better than the benchmark model & Decision tree's output using default parameters.

Refinement

I ran GridSearchCV with the following parameter values:

```
parameters = {'max_depth': [20, 50], 'min_samples_split' : [5, 10, 15]}
```

It returned the following estimator as the best one:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=50,
max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=15,
min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
oob_score=False, random_state=42, verbose=0, warm_start=False)
```

The above model gave an RMSE value of 3.29, which is significant improvement over the model with default parameters. I tried to further tune the parameters, but couldn't get better performance than this.

3.2.3 Support Vector Machines

I used SVR from sklearn.svm to implement this. By using the default parameters, I was not able to train the model. It was continuously running for more than 3 days on a system with 32 cores and 64 GB RAM. So I changed the max_iter to 10000 so that it can converge early. With this setting, I got an RMSE of 19.82 with other parameters as defaults.

Refinement

I ran GridSearchCV with the following parameter values:

```
parameters = {'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
'degree': [1, 2, 3, 4], 'epsilon': [0.1, 0.5, 1.5]}
```

It returned the following estimator as the best one:

```
SVR(C=1.0, cache_size=50000, coef0=0.0, degree=4, epsilon=1.5,
gamma='scale', kernel='poly', max_iter=10000, shrinking=True,
tol=0.001, verbose=False)
```

The above model gave an RMSE value of 19.74, which is better than the model with default parameters. But not a good choice as the other models gave far better performance.

3.2.4 AdaBoost Regression

I used AdaBoostRegressor from sklearn.ensemble module to implement this. By using the default base_estimator and default parameters, I got an RMSE value of 10.21, which is very high.

Refinement

I used the DecisionTreeRegressor obtained in Section 3.2.1 as the base_estimator of the AdaBoostRegressor. This resulted into an RMSE value of 4.57, which is far better than the default estimator.

I tried to run GridSearchCV to tune other parameters like learning_rate, but couldn't do so because of the resource constraints.

3.3 REFINEMENT

I used various improvements and techniques to improve the score and performance of the regressors:

- Ran GridSearchCV from sklearn.model_selection module to find the optimal value of the hyperparameters so that the model gives the best score (least value of RMSE). The corresponding value of the parameters for each regressor are discussed in section 3.2.
- Used n_jobs and pre_dispatch parameters in GridSearchCV to run a number of jobs in parallel on different CPU cores. It reduced the execution time significantly.
- Tried to use warm_start option in the regressors to make it re-use the solution of previous call to fit and use that to build upon on the new solution. But this did not

help much.

- Used several other options like increasing the cache_size, reducing the max_iterations to make the SVR converge early etc.

CHAPTER 4

RESULTS

4.1 MODEL EVALUATION AND VALIDATION

I ran the tuned models on both training as well as test data. The results are shown in Table 4.1.

Table 4.1: Results of different Models

Regressor	**Parameters**	**RMSE (Train Data)**	**RMSE (Test Data)**
DecisionTreeRegressor	max_depth=20 min_samples_leaf=20 min_samples_split=100	3.25	3.49
RandomForestRegressor	max_depth=50 min_samples_split=15	2.14	3.29
SVR	degree = 4 epsilon=1.5 kernel='poly' max_iter=10000	19.74	19.74
AdaBoostRegressor	base_estimator= DecisionTreeRegressor	3.95	4.57

As we can see from the table, RandomForestRegressor works best with the lowest RMSE value of 3.29. I will select this model as my final model.

4.2 JUSTIFICATION

For benchmark model, I have used a simple linear model given on Kaggle in the problem description of this completion. I compared the performance of other models with this model. The RMSE value of this model is given as to be 5.74184 on kaggle public score. And the RMSE value of the model selected by me is 3.29, which is much better than the benchmark model.

I got an R2 score of 0.89 which is pretty good as an R2 score closed to 1 is good.

CHAPTER 5

CONCLUSION

5.1 FREE-FORM VISUALIZATION

I drew a graph between the predicted values and actual results of the first 100 data points to compare the two. The results are pretty impressive as can be seen in Figure 5.1.

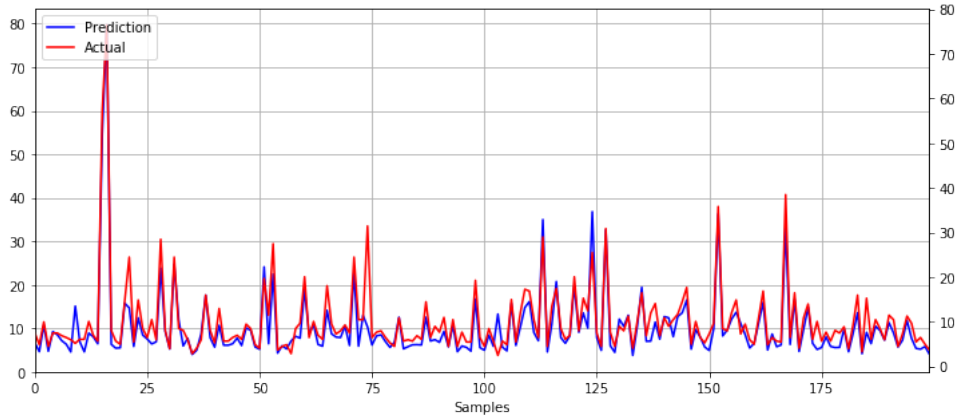


Figure 5.1: Predicted Values vs Actual Values

5.2 REFLECTION

This was an interesting problem. The main challenge was to handle large amount of data. Also the data contained a lot of outliers. So first I tried to remove as many outlier as I could think of. Then I ran a few regression algorithms on the data. With limited memory and processing power, the models were taking a large amount of time (in days) in fitting and prediction. GridSearchCV was also resource hungry.

First I tried to reduce the size of data by considering less number of rows from the original dataset. But It was not giving good enough results. Then I found that even though the data is large but my CPU is not being utilized to the fullest. Only 5-10% of CPU was being utilized. Then I got to know about n_jobs and pre_dispatch parameters in GridSearchCV to run a number of jobs in parallel. This increased the speed significantly. But still, some algorithms like SVR are too resource hungry that I had to reduce the amount of data to train.

5.3 IMPROVEMENT

One potential solution which I think will definitely improve the score of the model is to use more data to train the model. Currently I used only 5M rows out of a 55M row data.

There are several other factors that may affect the fare directly or indirectly e.g. special dropoff or pickup locations like Airports, roads having toll booths, traffic and population density of the place, days having special events or festivals etc. We can use these to engineer new features which will definitely improve the score of our model.

REFERENCES

[1] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 276–281, 2004.

[2] J. Van Lint, S. Hoogendoorn, and H. J. van Zuylen, "Accurate freeway travel time prediction with state-space neural networks under missing data," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5-6, pp. 347–369, 2005.

[3] R. Upadhyay and S. Lui, "Taxi fare rate classification using deep networks," *Transportation Research Part C: Emerging Technologies*, vol. 13, no. 5-6, pp. 347–369, 2005.

[4] Kaggle, "New york city taxi fare prediction." `https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/`, 2019.

[5] Kaggle, "New york city taxi fare prediction." `https://www.kaggle.com/c/new-york-city-taxi-fare-prediction#evaluation`, 2019.

[6] scikit-learn.org, "Decision trees." `https://scikit-learn.org/stable/modules/tree.html#tree`, 2019.

[7] scikit-learn.org, "Random forest regressor." `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`, 2019.

[8] dataaspirant.com, "Random forest algorithm." `http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/`, 2019.

[9] scikit-learn.org, "Support vector machines." `https://scikit-learn.org/stable/modules/svm.html`, 2019.

[10] Hackerearth.com, "Support vector machine." `https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/`, 2019.

[11] scikit-learn.org, "Adaboost regressor." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html, 2019.

[12] scikit-learn.org, "Boosting with adaboost." https://medium.com/diogo-menezes-borges/boosting-with-adaboost-and-gradient-boosting-9cbab2a1a 2019.

[13] Kaggle, "Simple linear model." https://www.kaggle.com/dster/nyc-taxi-fare-starter-kernel-simple-linear-model, 2019.

[14] Wikipedia, "Haversine formula." https://en.wikipedia.org/wiki/Haversine_formula, 2019.