PREDICTION OF TAXI FARE – NEW YORK CITY

Anisha Kunduru
Computer Science
Kennesaw State University
Kennesaw, Georgia, USA
akunduru@students.kennesaw.edu

Dinesh Kasanagottu
Computer Science
Kennesaw State University
Kennesaw, Georgia, USA
dkasanag@students.kennesaw.edu

Grishma Saparia
Computer Science
Kennesaw State University
Kennesaw, Georgia, USA
gsaparia@students.kennesaw.edu

Manogna Garlapati
Computer Science
Kennesaw State University
Kennesaw, Georgia, USA
mgarlap1@students.kennesaw.edu

Varun Gottam
Computer Science
Kennesaw State University
Kennesaw, Georgia, USA
vgottam1@students.kennesaw.edu

1. ABSTRACT

In comparison to any other city in the United States, taxis are utilized significantly more frequently in New York City. In New York City, approximately 200 million taxi rides are taken each year. Changes in traffic patterns, road closures, and large-scale events that attract a large number of New Yorkers may result from these enormous rides that are taken each month. Users of ride-hailing services like Uber, Lyft, and others can plan their trips in advance, which city taxi drivers do not have. Similar to other online taxi hailing apps, city taxi riders would greatly benefit from the ability to plan their rides in advance. We randomly selected 1.5 million Kaggle data in order to analyse previous taxi rides in New York City and predict the taxi fare. This includes the coordinates of the pickup and drop-off, the distance, the start time, the number of passengers, and the fare amount. In order to boost the efficiency of the machine learning models that were used, we added feature engineering to the initial dataset. The fare amount was predicted using KNN regression, Random Forest, Adaptive boosting, and SVM models.

2. RESEARCH STATEMENT AND CONJECTURE

There are many methods used to predict the rate using real-time data. But using real-time data takes more resources and the predictions are very slow. As Google introduced time estimates, computing fares became easy. So, we want to use previous data (data collected from taxis) to estimate the fares rather than real-time data. This helps in giving faster prediction and accuracy nearer to the real-time data predictions.

3. RELATED WORK

There are many reasons which could affect the prediction of taxi fare such as duration of travel, taxi fare might be higher during peak traffic hours, Weekday, weekend and specific hour of the day might impact fare amount, trip distance that is distance and the fare amount are directly proportional, neighbourhood impacts fare amount and airport pickups/drops.
One approach for predicting duration is by using real-time data collection to make individual predictions. The authors of [1] address the challenge by using GPS data from buses and a Kalman filter-based algorithm. [2] adopts a similar approach, using real-time data from smartphones installed inside vehicles. Highway travel time forecast yields better results than downtown area travel time prediction. This facilitates more precise predictions.

The authors of [3] predict travel time on congested freeways using a combination of traffic modelling, real-time data analysis, and traffic history. They attempt to dispel the myth that real-time analysis communication is instantaneous. Many other papers are also concerned with freeways. [4] predicts using Support Vector Regression (SVR), whereas [5] uses Neural Networks (SSNN). Predictive estimates of future transit times were introduced in the Google Maps API in 2015 [6]. This demonstrates the significance of being able to predict time travel without having real-time traffic data. This demonstrates the significance of being able to predict time travel without having real-time traffic data. By analysing data collected from taxis, we are attempting to solve a similar problem: estimating ride duration without real-time data. Being able to make such estimates would aid in making more accurate future predictions.

4. METHEDOLOGY

To solve the problem, we have implemented the following algorithms: Knn Regression, Random Forest, Adaptive boosting algorithm, Support Vector Regression.

4.1 PRE-PROCESSING

We randomly picked 10,000 trips as a training dataset of 1.5 million taxi rides in the Kaggle dataset. We also further divided the training data set into two parts: 80 percent for training and 20 percent for testing. The initial features of the dataset are as follows:

- pickup_datetime - timestamp value indicating when the taxi ride started.
- pickup_longitude - longitude coordinate of where the taxi ride started.
- pickup_latitude - latitude coordinate of where the taxi ride started.
- dropoff_longitude - longitude coordinate of where the taxi ride ended.
- dropoff_latitude - latitude coordinate of where the taxi ride ended.
- passenger_count - number of passengers in the taxi ride.
- fare_amount - dollar amount of the cost of the taxi ride.

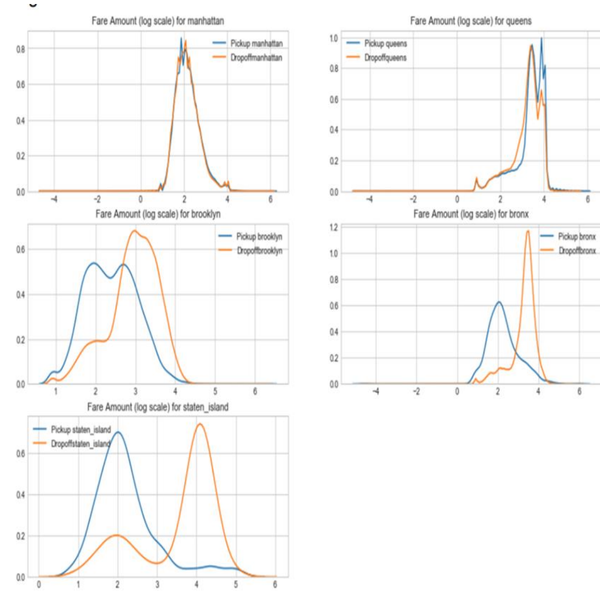
4.1.1 DATA CLEANING

After loading the data with Pandas, the first step was to clean up the data by removing rows with negative fare amounts because they don't seem to be realistic, removing any missing data, and noting passenger counts of more than six and zero for some taxi trips. And lastly dropped rows which are having pick and drop off coordinates out of New York city.

4.1.2 FEATURE ENGINEERING

Converted the pickup_datetime attribute of type Object to different primitive types such as pickup_date, pickup_day, pickup_hour, pickup_day_of_week, pickup_month, pickup_year using lambda functions, as we wanted to statistically compare how these features are affecting the taxi fare and total number of rides.

Calculated the haversine distance feature between the coordinates and padded as a distance feature.
We assessed whether our hypothesis of higher fare from certain neighborhoods are correct. Each pickup and drop off location were grouped through one of the five boroughs that make up New York City — Manhattan, Queens, Brooklyn, Staten Island, and the Bronx. And, yes, our hypothesis was proven correct for Manhattan, which had the majority of the pickups and drop offs, there was a difference in the pickup and drop off fare distribution for every other neighborhood. In addition, when compared to other neighborhoods, Queens had a higher mean pickup fare.



There is a high density of pickups near JFK and LaGuardia Airports. We then analyzed the average fare amount for pickups and drop-offs to JFK in reference to all trips in the train data and noticed that the fare was higher for airport trips. Based on this observation, we devised features to determine whether a pickup or drop-off was to one of New York's three airports: JFK, EWR, or LaGuardia.

And at last, for our Machine Learning model implementation, we considered the following 21 factors:

```
x_train.dtypes

pickup_longitude              float64
pickup_latitude               float64
dropoff_longitude             float64
dropoff_latitude              float64
passenger_count                 int64
pickup_day                      int64
pickup_hour                     int64
pickup_day_of_week              int64
pickup_month                    int64
pickup_year                     int64
trip_distance                 float64
pickup_borough                  int64
dropoff_borough                 int64
is_pickup_lower_manhattan       int64
is_dropoff_lower_manhattan      int64
is_pickup_JFK                   int64
is_dropoff_JFK                  int64
is_pickup_EWR                   int64
is_dropoff_EWR                  int64
is_pickup_la_guardia            int64
is_dropoff_la_guardia           int64
```

Data Before Feature Engineering

	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

Data After Feature Engineering

	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	pickup_day	pickup_hour	pickup_day_of_week	pickup_month
3945	-73.984912	40.736457	-73.984912	40.736457	2	28	3	0	3
4895	-73.971110	40.767340	-73.992218	40.749417	2	21	16	2	8
4656	-73.997551	40.741595	-74.000805	40.718606	1	24	15	2	3
6685	-73.948284	40.770464	-73.965706	40.752794	1	12	11	2	5
8809	-73.969140	40.754192	-73.980920	40.730688	2	29	1	2	12

pickup_year	...	pickup_borough	dropoff_borough	is_pickup_lower_manhattan	is_dropoff_lower_manhattan	is_pickup_JFK	is_dropoff_JFK	is_pickup_EWR
2010	...	2	2	1	1	0	0	0
2010	...	2	2	0	0	0	0	0
2009	...	2	2	1	1	0	0	0
2009	...	2	2	0	0	0	0	0
2009	...	2	2	0	1	0	0	0

is_pickup_EWR	is_dropoff_EWR	is_pickup_la_guardia	is_dropoff_la_guardia
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

4.2 KNN REGRESSION MODEL

KNN algorithm can be used for both classification and regression problems. This algorithm uses 'Feature similarity' to predict the values of any new data points. The concept of predicting the value of a new case is based on the K closest values to the similarity measure available. Finally, ours being regression, we choose mean of the values as final prediction.

a) Choosing similarity metric
Depending on the problem, we can use any similarity metrics such as Manhattan distance, Euclidean distance, cosine similarity etc. Here for implementing our model we have chosen Euclidean distance.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

b) Finding the optimal value of K
We plot the RMSE values for each K range and find that the best RMSE occurs when K is around 4-8. We picked a k-value of 4 because it gives us the best results.

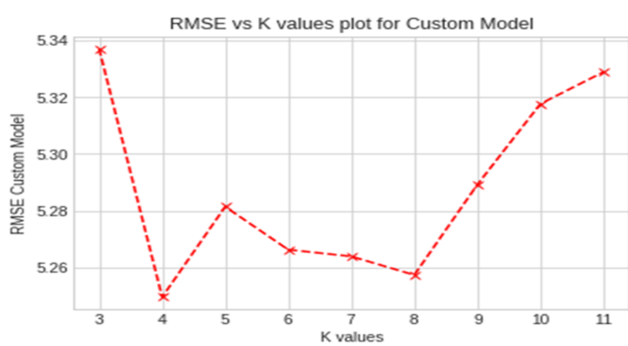


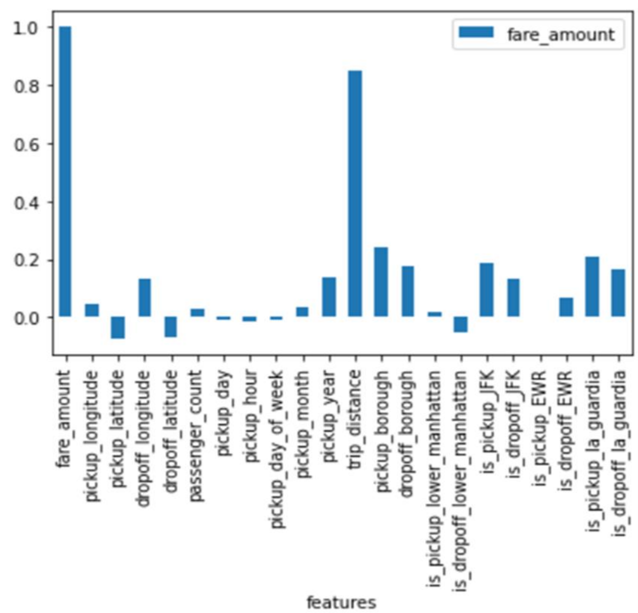
Fig: Plot of RMSE for finding K value using Custom model

c) Validation and Comparison with SkLearn
r2 metric score is used to examine the performance of the model.
Running the model with k = 4 the r2_score of the custom model is 0.677757322.
Running the sklearn model with k = 4 the r2_score of the sklearn model is 0.669399542.

4.3 RANDOM FOREST

Feature Elimination using Spearman correlation
A correlation coefficient measures the extent to which two variables tend to change together. This helps in better understanding how all the features in the dataset are related to fare amount in both strength and the direction of the relationship. We have used Spearman correlation to eliminate a few features which have the least correlation with fare amount to reduce the curse of dimensionality. This range lies between -1 to 1. So, eliminated 'is_pickup_EWR', 'pickup_day', 'pickup_day_of_week', 'pickup_month' columns as their values were close to 0.

Below is the graph that plots all the features and their corresponding spearman correlation values with respect to 'fare_amount'.

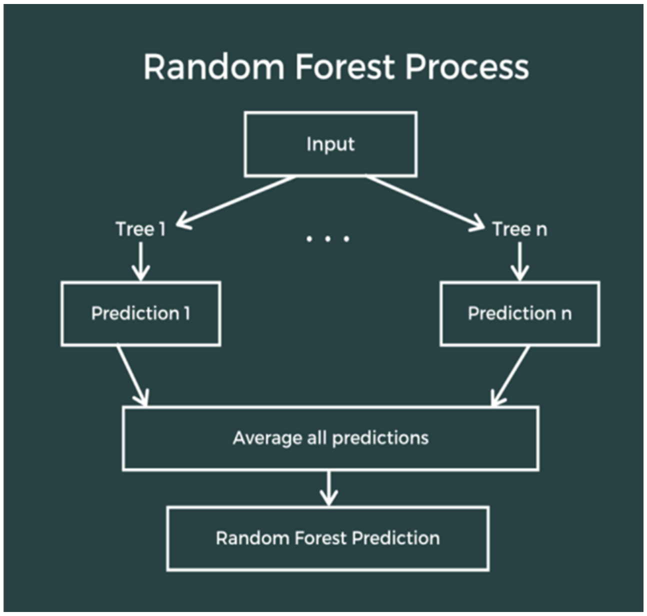


DECISION TREE REGRESSION
The data is partitioned into subsets that contain instances with similar values (homogenous) and a tree is built top-down from a root node. Decision trees can handle both categorical and numerical data. The leaf nodes represent a decision on the numerical target and the topmost decision node in a tree corresponds to the best predictor called the root node. We used standard deviation to calculate the homogeneity of a numerical sample. This process is run recursively until all data is processed or stopping criteria [*min_samples_split* and *max_depth*] are met.

$$\textit{Standard Deviation} = S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

RANDOM FOREST REGRESSION

It is a supervised learning algorithm that uses **ensemble learning** technique for regression i.e., combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. Random Forest uses bagging for building decision trees. In bagging (also called bootstrap aggregating), multiple trees are created, and the final output is an average of all the different outputs predicted by multiple decision trees. Bootstrap samples of data are taken and each sample trains a weak learner which helps to reduce the high variance which is usually seen in decision trees.



EVALUATION OF MODEL

Used R2 score and RMSE to evaluate the model. As observed, there was a significant difference and improvement in the prediction of the model after applying the ensemble bagging technique - Random Forest over decision tree.

Decision Tree Result

max_depth	*min_samples_split*	R2 score Sklearn model	R2 score Custom model	RMSE Sklearn model	RMSE Custom model
2	2	0.4273056594378327	0.3238749853762644	7.8297984582837906	7.5762738647324
3	3	0.6078414592209112	0.6738237451027293	5.079722005575151	4.752681021870103
5	3	0.4627056676932904	0.5181555662842368	6.258676020348406	6.038751232044826

Random Forest Result

n_estimators	R2 score Sklearn model	R2 score Custom model	RMSE Sklearn model	RMSE Custom model
10	0.8096418417112098	0.7799057508379961	3.6389265290986827	3.9443968904628615
50	0.7568671453209761	0.6637709922784956	4.425487183586003	5.207655728431738
100	0.51762978743512304	0.589798376472653	6.4254871835890135	6.987467268723432

4.4 ADAPTIVE BOOSTING ALGORITHM

Using pearson coefficient for correlation between two variables, pickup_longitude, pickup_latitude, drop-off_longitude, drop-off_latitude, pickup_day, and passengaer_count is removed to increase the model's performance. This is an ensemble learning where weaker models are trained sequentially to make them stronger models for more accurate predictions. Generally, while using Adaboost, we use Decision Tree Regressor.

Adaboost is a boosting technique used to boost the weak learner and make it to strong learner.Regression in Adaboost algorithm follows below steps. Initially, to each training pattern we assign a weight wi=1/N.The probability that training sample i is in the training set is pi=wi /Σwi where the Σwi is summation over all members of the training set. Construct a regression machine t from that training set. Each machine makes a hypothesis. Pass every member of the training set through this machine to obtain a prediction. Calculate a loss for each training sample and find its average. Find β=L/1−L, where L is the average loss. Update the weights: wi→wiβ**[1−Li] where Li is the loss of each sample. Take these sample weights and pick the values from training set as probabilities and perform above steps repeatedly. After performing these steps iteratively, find the weighted median of all the results from estimators. For a sample i, if its loss is more than it's updated weight will be less, which will make its probability to be picked up on the next iteration will be decreased. In our experiment we take weak learner like Decision Tree regressor and boost the decision tree regression to get better results.

We take decision trees implemented from scratch and form scikit library. Here number of estimators are number of weaker models in sequential order.

No:of estimators	R2 score of ADABOOST SCRATCH	R2-score of Adaboost scikit learn
50	0.7829	0.7297
100	0.7874	0.72467
150	0.7860	0.717897

Scores of the decision trees after adaboost. As we can see in the previous result, the R2 score did not increase much after 50 estimators.

Regressor on which boost applied.	R2-score of adaboost implemented from scratch	R2-score of adaboost from scikit-learn implementation
Decision Tree regressor Scratch	0.44	-
Scikit learn Decision Tree regresor	0.612	0.6639

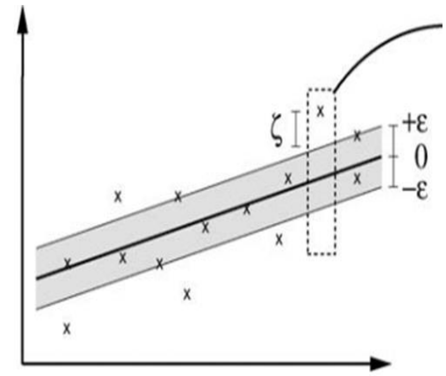
4.5 SUPPORT VECTOR REGRESSION

Feature Elimination:
Because the Pearson correlation coefficient between 'taxi fare' and these features is not very significant, we deleted 'pickup EWR','passenger count','pickup day','pickup hour', and 'pickup month' from our list of features. Aside from those, we evaluated model using all of the remaining features. We can see the Pearson coefficient values from the list attached. We implemented linear support vector regression without any Kernel.

```
fare_amount                     1.000000
pickup_longitude                0.004688
pickup_latitude                -0.004640
dropoff_longitude               0.006379
dropoff_latitude               -0.003678
passenger_count                 0.013600
pickup_day                     -0.000554
pickup_hour                    -0.018136
pickup_day_of_week             -0.011963
pickup_month                    0.022897
pickup_year                     0.115301
trip_distance                   0.043485
pickup_borough                  0.455330
dropoff_borough                 0.334012
is_pickup_lower_manhattan      -0.042811
is_dropoff_lower_manhattan     -0.083331
is_pickup_JFK                   0.408714
is_dropoff_JFK                  0.324005
is_pickup_EWR                   0.053971
is_dropoff_EWR                  0.222243
is_pickup_la_guardia            0.279979
is_dropoff_la_guardia           0.226536
```

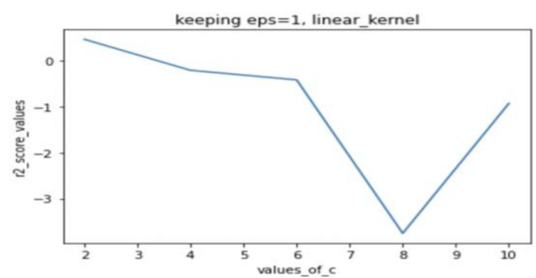
LINEAR SUPPORT REGRESSION

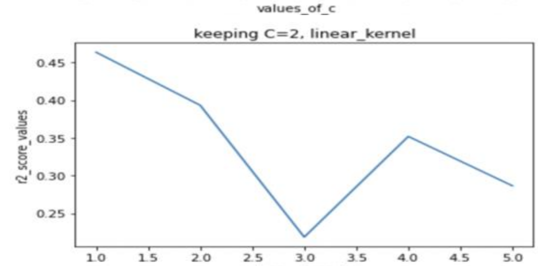
For implementation of linear support vector regression, we used gradient descent to find the optimal values of ‘w’ and ‘b’.with these optimal values of w and b I predicted the y values for x_test.I got r2_score as -0.22 when regularization constant ‘C’=10.The reasons for the less value of r2 score is: The dimension of the data I used is 16 which will very hard for linear svr to perform well because i didn’t use any kernel functions.When i am increasing the ‘C’(regularization parameter) value i am getting the better r2 score as we can see in the below graph.I kept epsilon value constant (eps=1) when varying the ‘C’.



LINEAR KERNAL

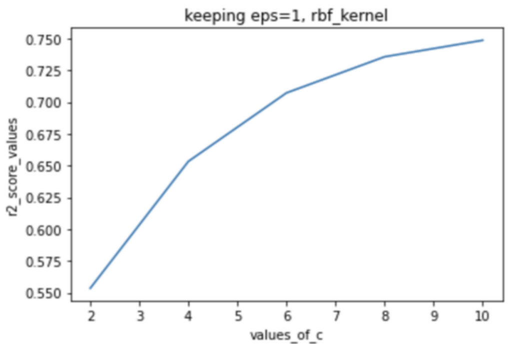
After performing linear svr without any kernels then we implemented svr with linear kernel using scikit-learn implementation. with scikit-learn implementation I got good r2_score=0.47 when C=2 but for some values of C we also got negative r2 score. As a conclusion, we can deduce that even the linear kernel does not perform adequately. For many C values, my custom model outperformed the scikit-learn linear kernel implementation.

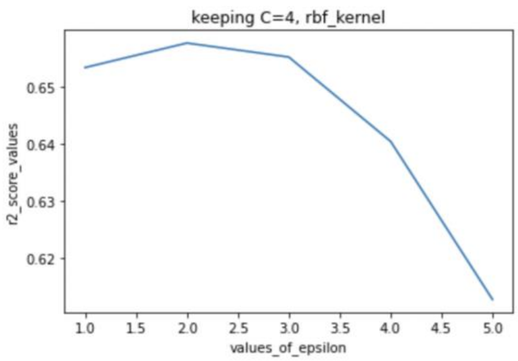




RBF KERNAL

We implemented the rbf kernel using scikit learn. For the rbf kernel we fixed gamma value=’auto’ and then changed the remaining hyperparameters. We can clearly see that for C=10 we are getting r2_score of value 0.75 which is the best score on this dataset. We can see that r2_scores are increasing when 'C' is increasing. With 'epsilon=1', we changed inverse regularization parameter C then calculated r2_score values. When we fixed the 'C' value as 4 and changed the epsilon values. We can see that with increase of epsilon value the r2_score is slightly decreasing.





5. EXPERIMENTAL RESULTS

5.1 DIMENSIONAL REDUCTION

Apart from the feature engineering discussed in the above section, we also performed feature elimination using spearman coefficient and pearson coefficient, to compute the correlation between fare price and all other features and dropped a few features whose correlation values were very low in adaptive boosting algorithm and random forest algorithm respectively. This made the models comparatively more accurate and could also notice significant improvement in performance.

5.2 ACCURACY

In order to evaluate the models on test data we used R2_score and rmse to check the accuracy.In addition to that we also used Sklearn models to draw the comparison with the custom models built.As observed in the table below.

MODEL	SKLEARN R2	CUSTOM MODEL R2	SKLEARN RMSE	CUSTOM MODEL RMSE
KNN	0.66939	0.67757	-	-
Random forest	0.80964	0.77990	3.63892	3.94439
Adaboost	0.72467	0.7874	4.37736	4.98229
SVR	0.75 for rbf kernel. 0.4 for linear kernel.	-0.22 for svr without any kernel	-	-

6. CONCLUSION

It is observed that out of all the models Random Forest performed better when compared to other models with an R2-score of 0.779 and 0.8 accuracy of Sklearn model.

We got to the notion that the trip distance was the most essential factor in deciding the fare amount, While the number of passengers was the least important one. After Feature engineering, the R2_score decreased significantly

Future work:We investigated few additional models such as LightGBM,neural networks and XGBM.We can improve the performance even further by using these models.

7. REFERENCES

[1] Vanajakshi, L., S. C. Subramanian, and R. Sivanandan. "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses." IET intelligent transport systems 3.1 (2009): 1-9.

[2] Biagioni, James, et al. "Easytracker: automatic transit tracking, mapping, and arrival time prediction using smartphones." Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems. ACM, 2011.

[3] Yildirimoglu, Mehmet, and Nikolas Geroliminis. "Experienced travel time prediction for congested freeways." Transportation Research Part B: Methodological 53 (2013): 45-63.

[4] Wu, Chun-Hsin, Jan-Ming Ho, and Der-Tsai Lee. "Travel-time prediction with support vector regression." IEEE transactions on intelligent transportation systems 5.4 (2004): 276-281.

[5] Van Lint, J. W. C., S. P. Hoogendoorn, and Henk J. van Zuylen. "Accurate freeway travel time prediction with state-space neural networks under missing data. "Transportation Research Part C: Emerging Technologies 13.5 (2005): 347-369.

[6] Kelareva, Elena. "Predicting the Future with Google Maps APIs." Web blog post. Geo Developers Blog, https://maps-

apis.googleblog.com/2015/11/predicting-futurewith-google-maps-apis.html Accessed 15 Dec. 2016.

[7] SKlearn resources for decision tree and random forest concepts:
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
https://scikit-learn.org/stable/modules/tree.html#tree
https://scikit-learn.org/stable/modules/ensemble.html#forest

[8] Tuning a decision tree
https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680

[9] ID3 algorithm
https://guillermoarriadevoe.com/blog/building-a-id3-decision-tree-classifier-with-python

[10] Scikit Learn Implementation of Regression Tree and Random forest
https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/tree/_classes.py#L1034
https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/ensemble/_forest.py#L1399

[11] Understanding support vector regression
https://www.youtube.com/watch?v=ECXc2P6t3TY

[12] Scikit learn implementation of linear kernel
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[13] Scikit learn implementation of svr kernel
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html