CS 235 Project Report

TAXI FARE PREDICTION - NEW YORK CITY

Roshini Angamgari 862325068 ranga003@ucr.edu
Aishwarya Pagadala 862324772 apaga002@ucr.edu
PrudhviManukondaTalluri 862325916 pmanu005@ucr.edu
Ghannesh chandra 862325413 gtall004@ucr.edu

1. Introduction

Taxis are used much more frequently in New York City than in any other city in the United States. Every year, approximately 200 million taxi rides are taken in New York City. These humongous rides taken each month can provide changes in traffic patterns, road closures, and large-scale events that draw a large number of New Yorkers. Ride-hailing companies such as Uber, Lyft, and others allow users to plan their trips ahead of time, which city taxi drivers do not have. It would be extremely beneficial for city taxi riders if they could plan their taxi rides ahead of time, just like they can with other online taxi hailing apps.
In order to predict the taxi fare by analyzing previous taxi rides in New York City, we randomly picked up 1.5M data from kaggle. This includes pickup and dropoff coordinates, trip distance, start time, number of passengers, and fare_amount. We further applied feature engineering for the initial dataset to improve the performance of the machine learning models used. KNN regression, Random forest, Adaptive boosting and SVM models were used to predict the fare amount.

2. 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.

3. Proposed Methods

To solve the problem we have implemented the following algorithms:
KNN Regression - Roshini Angamgari
Random Forest - Aishwarya Pagadala
ADAPTIVE boosting algorithm - Prudhvi Manukonda
SVM - Talluri Ghannesh chandra

3.1 Knn Regression – Roshini Angamgari

3.1.1 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.

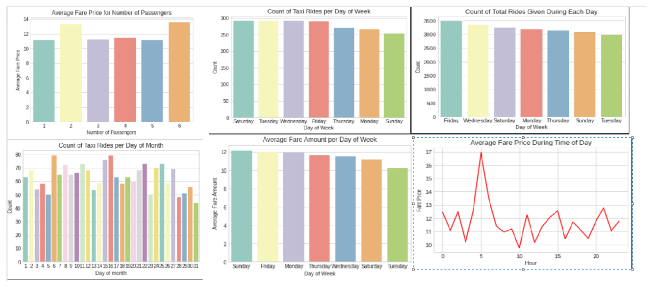
3.1.2 Cleaning

Used pandas to load the data and did initial cleaning of the data like dropping rows which have negative fare amount as it doesn't seem to be realistic, any missing data, noticed passenger count which is greater than six and having zero passenger count for some of the taxi trips. And lastly dropped rows which are having pick and drop off coordinates out of new york city.

3.1.3 Feature engineering

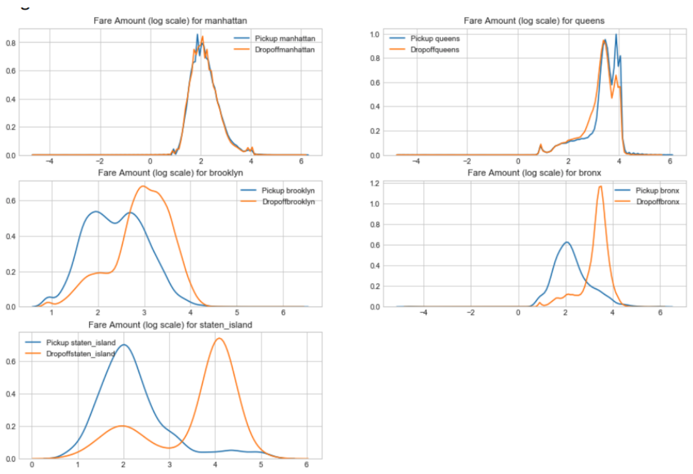
Converted the pickup_datetime attribute of type Object to different primitive types such as pickup_date, pickup_day, pickup_hour, pickup_day_of_week, pickup_month, pickup_year using lambda functions, as we wanted to statistically compare how these features are affecting the taxi fare and total number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

After conversion it is observed that the fare amount is at its highest at 5 a.m., according to the graphs below. And, during the analysis, we discovered that the greatest number of trips are to/from the airport at 5 a.m., and that the number of pick-ups are higher on Saturday, but the fare amount is still relatively low on Saturday. On Sunday and Monday, the number of trips is lower, but the fare is relatively high. Also, based on the graph below, we can deduce that taxi demand is highest at 7 p.m. and lowest at 5 a.m., assuming that a constant number of taxis are available at all times throughout the day.



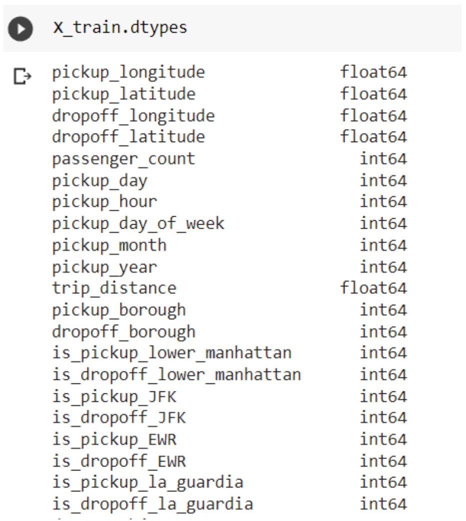
Calculated the haversine distance feature between the coordinates and padded as a distance feature.

We assessed whether our hypothesis of higher fare from certain neighborhoods is correct. Each pickup and drop off location was 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.



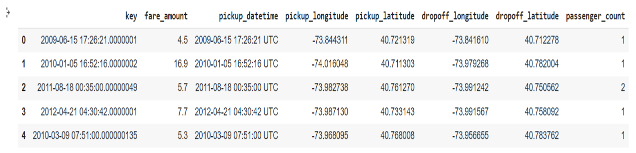
There is a high density of pickups near JFK and LaGuardia Airports. We then analyzed the average fare amount for pickups and drop-offs to JFK in reference to all trips in the train data and noticed that the fare was higher for airport trips. Based on this observation, we devised features to determine 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:



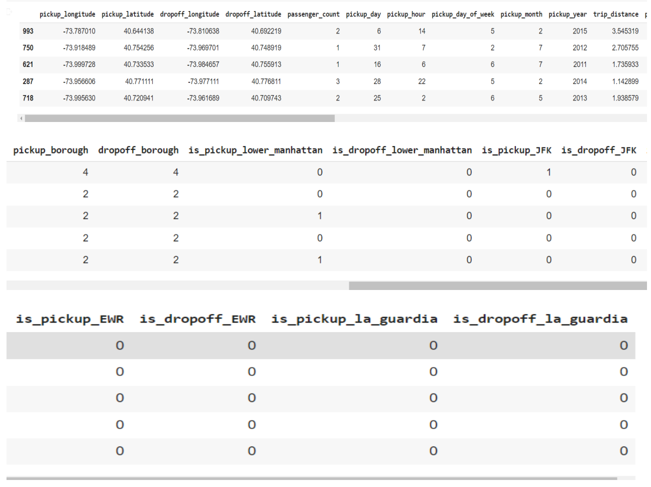
```
X_train.dtypes

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	pickup_year	trip_distance
993	-73.787010	40.644138	-73.810638	40.692219	2	6	14	5	2	2015	3.545319
750	-73.918489	40.754256	-73.969701	40.748919	1	31	7	2	7	2012	2.705755
621	-73.999728	40.733533	-73.984657	40.755913	1	16	6	6	7	2011	1.735933
287	-73.956606	40.771111	-73.977111	40.776811	3	28	22	5	2	2014	1.142899
718	-73.995630	40.720941	-73.961689	40.709743	2	25	2	6	5	2013	1.938579

pickup_borough	dropoff_borough	is_pickup_lower_manhattan	is_dropoff_lower_manhattan	is_pickup_JFK	is_dropoff_JFK
4	4	0	0	1	0
2	2	0	0	0	0
2	2	1	0	0	0
2	2	0	0	0	0
2	2	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

3.1.4 KNN Regression Model Implementation:

KNN is a non-parametric supervised machine learning model that saves all available data and predicts future instances using a similarity metric of your choice. The concept of predicting the value of a new case based on the K closest values to the similarity measure available. It can be used to solve problems involving classification and regression. It is an example of a non-parametric method, where we are free to learn any mapping function based on the training data and do not assume any preset form of learning function.

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.

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

When a new test data point is supplied, the KNN model maintains all of the training data set, and the goal is to select K points from the training set that are closest to the test data point. So we're looking for the spots that have the lowest similarity metric.

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.

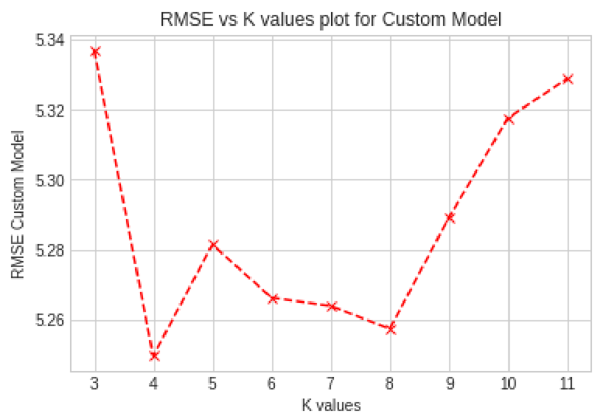


Fig : Plot of RMSE for finding K value using Custom model

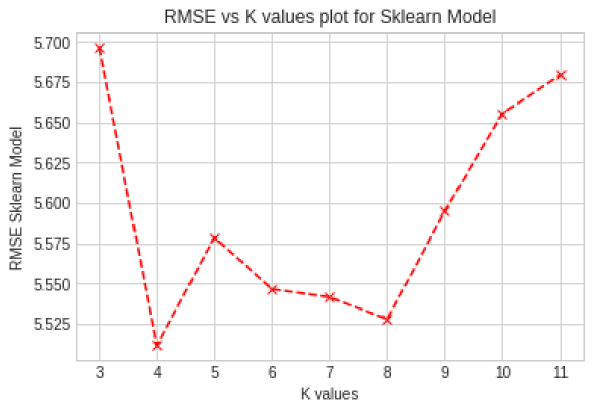
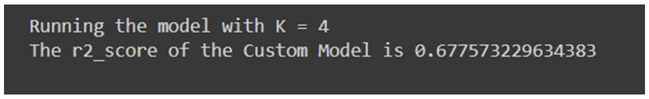


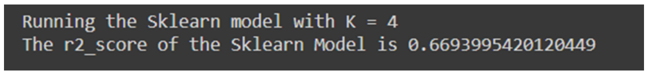
Fig : Plot of RMSE for finding K value using Sklearn model

c)Validation and Comparison with SkLearn

The r2 metric score is used to examine the performance of the model. This is a built-in metric of Sklearn and is a standard criterion for evaluating model accuracies. The r2 values for the Sklearn KNeighborsRegressor() model and the custom KNNRegression model are as follows.



```
Running the model with K = 4
The r2_score of the Custom Model is 0.677573229634383
```



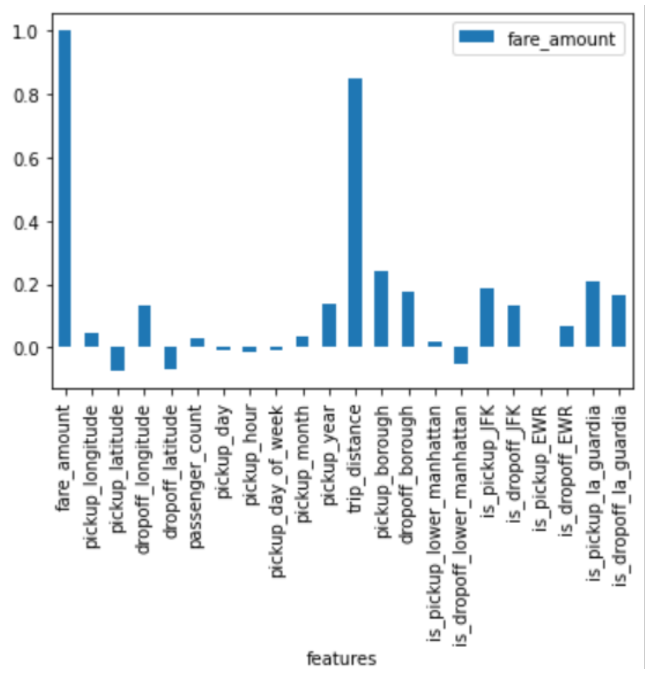
```
Running the Sklearn model with K = 4
The r2_score of the Sklearn Model is 0.6693995420120449
```

3.2 Random Forest - [Aishwarya Pagadala]

3.2.1 Feature Elimination

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. I used Spearman correlation to eliminate a few features which have the least correlation with fare amount to reduce the curse of dimensionality. The Spearman correlation evaluates the monotonic relationship between two continuous or ordinal variables. In a monotonic relationship, the variables tend to change together, but not necessarily at a constant rate. This range lies between -1 to 1 and I have eliminated 'is_pickup_EWR', 'pickup_day', 'pickup_day_of_week', 'pickup_month' columns as their value was close to 0.

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



After dropping the above-mentioned columns from the dataset, the final dataset was split into training data: 80% and test data:20% for 10k rows of our dataset (as executing on the custom model on huge data was taking too long) and tested with decision tree model and random forest model.

3.2.2 Decision Tree Regressor

A decision tree builds regression or classification models in the form of a tree structure. The data is partitioned into subsets that contain instances with similar values (homogenous) and a tree is built top-down from a root node. The final result is a tree with decision nodes and leaf nodes. These leaf nodes represent a decision on the numerical target. The topmost decision node in a tree corresponds to the best predictor called the root node. Decision trees can handle both categorical and numerical data. We use standard deviation to calculate the homogeneity of a numerical sample.

3.2.2.1 Methodology

After studying the scikit learn documentation and developing an in-depth understanding of the Decision trees algorithm, I have programmatically implemented a generalized Decision tree class in python using NumPy. Parameters used in constructing custom decision tree are max_depth and min_samples_split. min_samples_split is the minimum number of samples required to split an internal

node. max_depth is the maximum depth of the tree. If None is given as input, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
Basically, the underlying algorithm for building decision trees is ID3, which employs a top-down, greedy search with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with Standard Deviation Reduction.

First, we find standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

$$Count = n \qquad Average = \bar{x} = \frac{\sum x}{n}$$

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

The dataset is then split into different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

$$SDR(T,X) = S(T) - S(T,X)$$

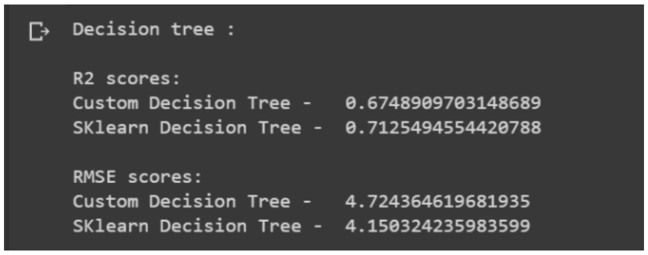
The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. The attribute with the largest standard deviation reduction is chosen for the decision node (i.e., the most homogeneous branches). The dataset is divided based on the values of the selected attribute. This process is run recursively until all data is processed or stopping criteria are met.

3.2.2.2 Evaluation of Custom decision tree regressor with SKlearn model

Used 10k rows from the dataset and experimented with the parameters to determine the best max_depth and min_samples_split values.

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

With the above results, as observed the best after max_depth = 3 the accuracy started decreasing. Similarly, after trying out different values for min_samples_split, the results were best for value 2. So in order to reduce the overfitting to the training data, the best parameters for the decision tree came out to be min_samples_split = 3 and max_depth =3 for this sample of data.


```
Decision tree :

R2 scores:
Custom Decision Tree -   0.6748909703148689
SKlearn Decision Tree -  0.7125494554420788

RMSE scores:
Custom Decision Tree -   4.724364619681935
SKlearn Decision Tree -  4.150324235983599
```

Now, these results with the decision tree model will be used to compare the accuracy of the random forest model in order to see how bagging helps with the accuracy of the predictions.

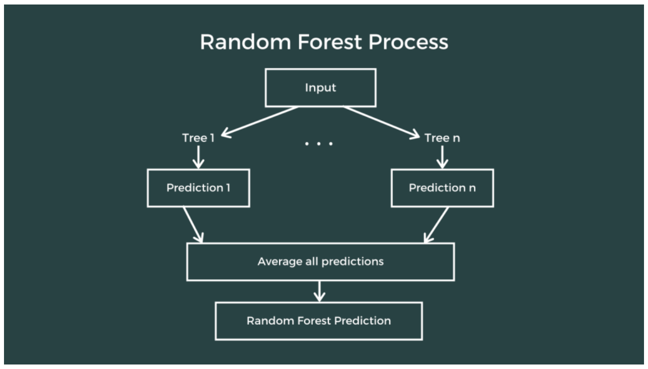
3.2.3 Random Forest Regression

It is a supervised learning algorithm that uses ensemble learning techniques for regression. Ensemble learning method is a technique that 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 models are created, and the final output is an average of all the different outputs predicted by multiple decision models. Here, bootstrap samples are taken and each sample trains a weak learner. This technique is mainly used to reduce the high variance which is usually seen in decision trees.

3.2.3.1 Methodology

Implemented a custom Random forest model which uses the above-mentioned custom Decision tree class, but here multiple trees will get trained on bootstrapped subsets of the training data set, make predictions with each of the trees, and finally compute the mean to decide the final prediction resulting in lower bias fitting training data more efficiently to improve the predictive accuracy and control overfitting.
The following are the parameters used in our random forest: n_estimators, min_samples_split,
max_depth. Here, n_estimators are the number of trees in the forest. (The other two parameters are explained in the decision tree section.)
Here, We first take the number of trees we want to build from the input, then perform bootstrap sampling on the data with replacement and build decision trees separately. the

trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

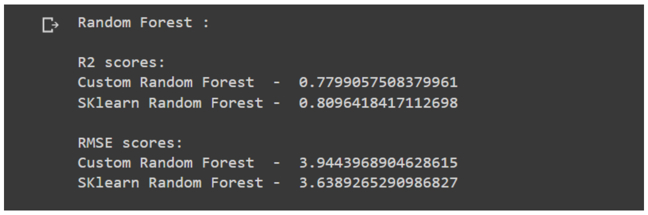


3.2.3.2 Evaluation of Custom Random forest regressor with SKlearn model

Used 10k rows from the dataset and experimented with the parameters to determine the best number of trees i.e n_estimators. I have used min_samples_split = 2 and max_depth = 5 for this sample of data based on above observations. It took 3-6 hrs of runtime to execute each scenario.

n_estimators	R2 score Sklearn model	R2 score Custom model	RMSE Sklearn model	RMSE Custom model
10	0.809641841711298	0.7799057508379961	3.6389265290986827	3.944396890462 8615
50	0.7568671453209761	0.6637709922784956	4.425487183586003	5.207655728431738
100	0.51762978743512304	0.589798376472653	6.4254871835890135	6.987467268723 432

As observed in the above table, the model gave best results when the number of trees was 10.

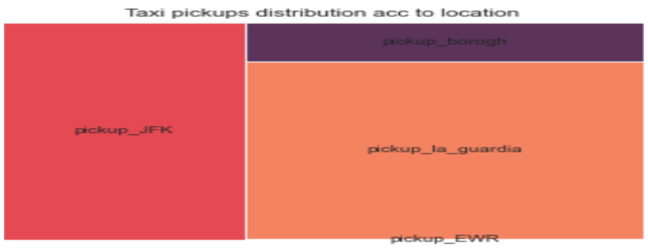


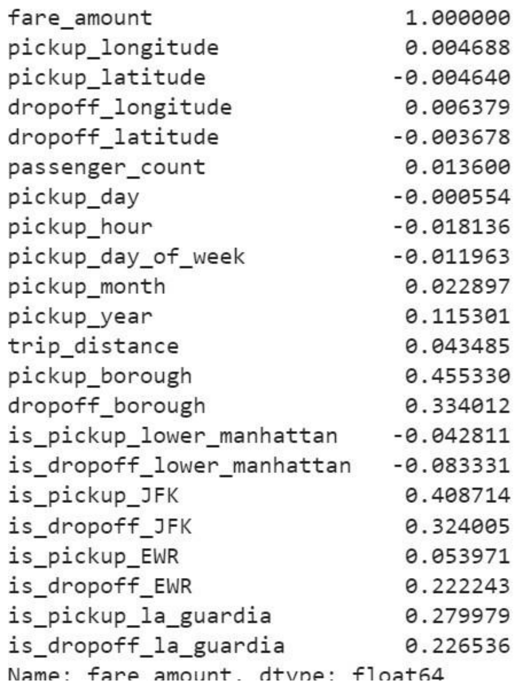
```
Random Forest :

R2 scores:
Custom Random Forest  -  0.7799057508379961
SKlearn Random Forest -  0.8096418417112698

RMSE scores:
Custom Random Forest  -  3.9443968904628615
SKlearn Random Forest -  3.6389265290986827
```

3.3 ADAPTIVE Boosting Algorithm- [Prudhvi Manukonda]

3.3.1 Feature Elimination

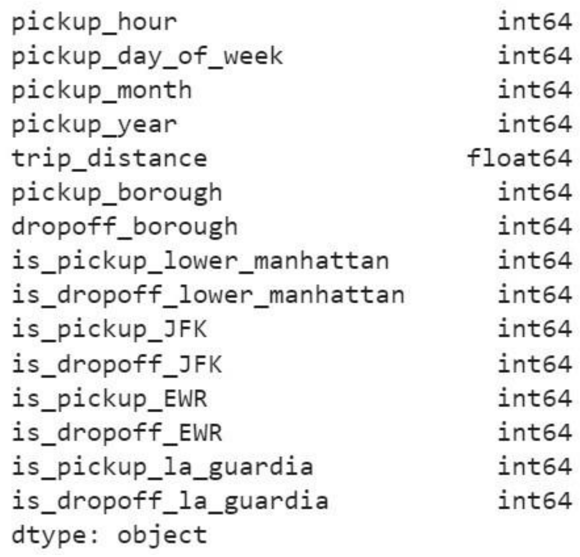
In order to get a better visualization,I have plotted tree map with location pickups as labels in tree map and values as number of rides at that pickup location.Also, In order to reduce the curse of dimensionality,I have calculated the basic pearson correlation coefficient between taxi fare and all features and eliminated the features which have very low values.Below attached is the correlation coefficients for corresponding features. Based on the corresponding coefficient values removed feature like,pickup_longitude,pickup_latitude,drop-off_longitude,dro-p_offlatitude,pickup_day,passenger_count.Below attached correlation coefficients and Tree map .





```
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
Name: fare_amount, dtype: float64
```

Final data features used for training after reducing the dimensions attached below.


```
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
dtype: object
```

3.3.2 Overview of Adaboost algorithm in regression

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 it's loss is more than it's updated weight will be less, which will make it's 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.

3.3.3 RMSE Calculation and Comparison with Scikit learn

Firstly, I took 1000 samples and performed train_test_split on feature engineered dataset and implemented decision tree from scratch.Trained Decision tree on training data and predicted for test data and noted R2-score values of scratch implementation and scikit learn implementation.

Regressor	R2-score
Decision Tree regressor	0.24
Decision Tree from Scratch	0.24
Decision Tree regressor SCIKIT learn	0.42

After implementing decision tree scratch,implemented adaboost on it and compared it with scikit learn values.

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

After running on the data I understood that running decision tree on whole dataset is quite time consuming so,took large dataset of 99999 samples and took scikit learn implemented decision tree regression and applied boosting algorithm on it and found r2-score for different values of estimators,It seemed like number of estimators after 50 didn't made a crucial role in increasing r2-score.
SCIKIT learn decision tree on bulk data set got r2 score of

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

We are going to measure classifier performance based on R2-score of classifier we can see that basic decision tree has R2-score of 0.44 we boosted and improved the performance of decision tree regression to 0.78 R2 score.In this model main hyper parameter is No:of estimators we can see that there is no significant impact on increasing above 50 estimators on model's performance.Our implementation and scikit learn implementation ,Almost performed equally.

3.4 Support vector regression-Ghannesh Talluri

3.4.1 Feature Elimination

I removed 'is_pickup_EWR','passenger_count','pickup_day','pickup_hour','pickup_month'from my set of features because the Pearson correlation coefficient between 'taxi fare' and these features are not that considerable.Other than those I considered all remaining features.

3.4.2 Data division

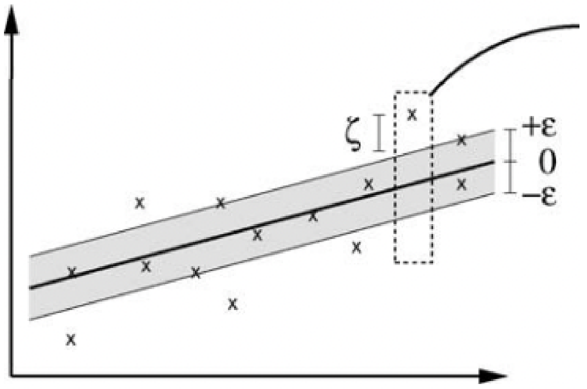
I used k-fold cross validation for splitting the data.I split the data into 5 parts(i.e,k=5).I used four sets for training the model and one set for testing the model.

3.4.3 Overview of epsilon support vector regression

$$|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases}$$

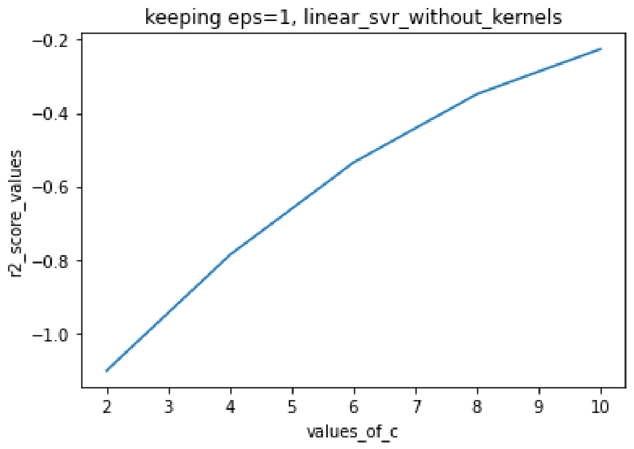
$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ \text{subject to} & \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{array}$$

This is the loss function i used.For the points which satisfies this condition |y_actual-(wx+b)|<=epsilon then those points lie inside the margin.The points which don't satisfy this condition lie outside of the margin . We don't take any penalty for the points that lie inside the margin.slack variable 'eta' is defined as the difference between the margin and the point. 2 epsilon value is the width of the boundary .We can see these variables clearly in the below plot.



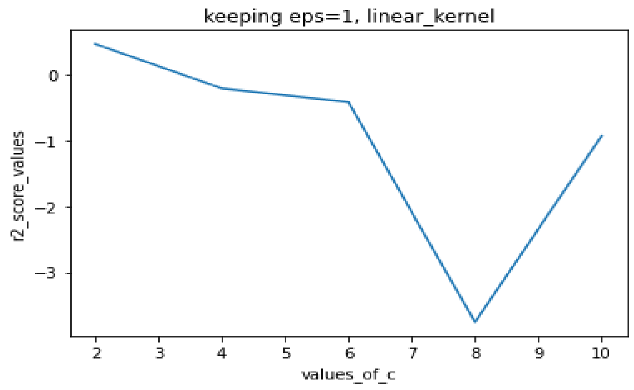
3.4.4 Implementation of linear support vector regression

For implementation of linear support vector regression I 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 be 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'.

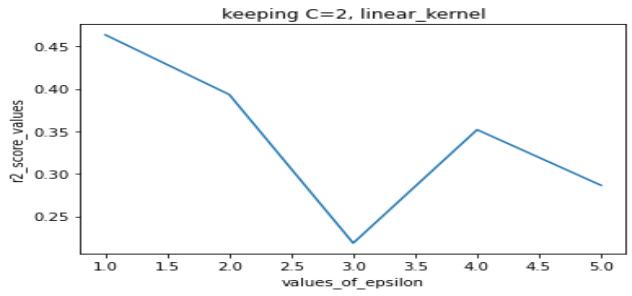


3.4.5 Linear kernel

After performing linear svr without any kernels then I 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 I also got negative r2 score (like -0.2 for C=4, -0.4 for C=6 and -3.75 for C=8) .So,we can conclude that even linear kernel is not performing well.For some values of C my custom model performed better than scikit-learn implementation of linear kernel.
For a linear kernel We have two parameters, epsilon,C. I plotted the graphs for different values of 'C' vs r2_score by fixing epsilon=1.

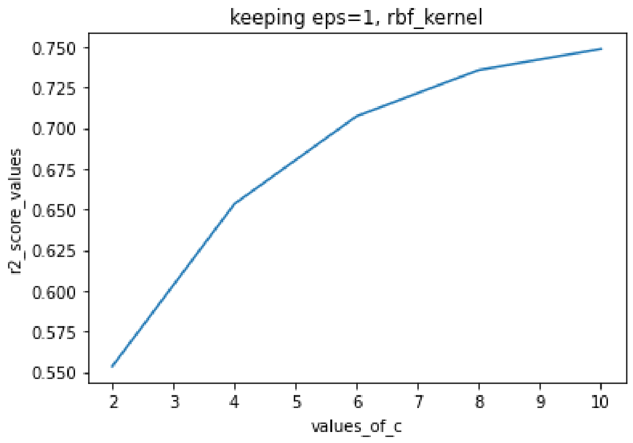


I also fixed value of C=2 and then varied epsilon then r2_score values are:

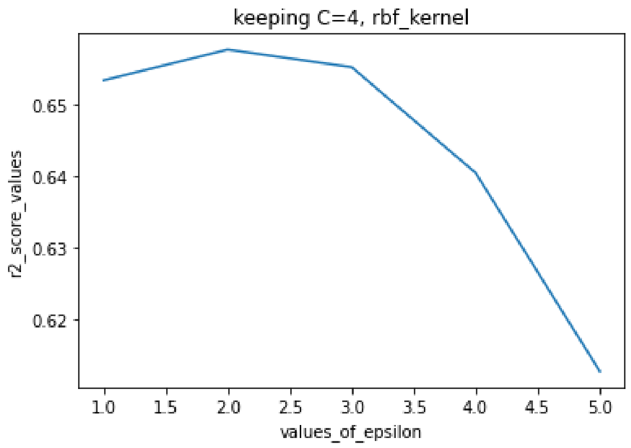


3.4.6 RBF kernel

I implemented the rbf kernel using scikit learn. For the rbf kernel I fixed my gamma value='auto' and then changed the remaining hyperparameters.
With 'epsilon=1', I changed the inverse regularization parameter C then calculated r2_score values.



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.Next,I fixed the value of C=4 and changed the different epsilon values then i got r2_score values as:



Based on the r2_score We can finally conclude that the rbf kernel is the best model for the given dataset.we got a good r2_score for every value of 'C'.

4. Experimental Evaluation

4.1 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.

4.2 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	-	-

4.3 Performance

We tested the performance of our custom models and Sklearn models on 10k trips of the dataset sorted by pickup_datetime, which is the timestamp value indicating when the taxi ride started. KNN took around 45 minutes to execute. Random forest and Adaboost took a very long time to run initially, but after performing feature elimination with the help of Spearman coefficient and Pearson coefficient the run time significantly decreased. Yet, it took around 3 to 7 hours. In contrast, SVM took very less time and was executed quickly.

5. Discussion and Conclusions

It is observed that out of all the models Random Forest performed better when compared to other models used with an accuracy of 0.779 with custom model and 0.8 accuracy for 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 RMSE decreased significantly. In addition, we discovered that fare amount and demand were inversely proportional.

6. Future Work

We learned the significance of feature engineering by breaking down the project into small chunks and performing microanalysis. This resulted in a reduction in error. We also investigated a few additional ML models such as Light GBM, neural networks, and so on. We intend to improve performance even further by using neural networks to tune hyper-parameters and build accurate 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-future-with-google-maps-apis.html Accessed 15 Dec. 2016.

7.1. Data Set:

https://www.kaggle.com/c/new-york-city-taxi-fare-prediction

7.2 Calculate distance between locations:

https://www.travelmath.com/flying-distance/

7.3 KNN:

a. Shichao Zhang, Ming Zong, Xiaofeng Zhu, and Debo Cheng Learning (2017) k for kNN Classification ACM Transactions on Intelligent Systems and Technology. https://dl.acm.org/doi/epdf/10.1145/2990508

b.Distance metrics and K-Nearest Neighbor (KNN) https://medium.com/@luigi.fiori.lf0303/distance-metrics-and-k-nearest-neighbor-knn-1b840969c0f4

c. A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python code) - https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

7.4 Random Forest:

1.SKlearn resources for decision tree and random forest concepts
a.https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html
b.https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
c.https://scikit-learn.org/stable/modules/tree.html#tree
d.https://scikit-learn.org/stable/modules/ensemble.html#forest
2.Pearson and Spearman correlation methods
https://towardsdatascience.com/clearly-explained-pearson-v-s-spearman-correlation-coefficient-ada2f473b8
3.Tuning a decision tree-
https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680
4.ID3 algorithm
https://guillermoarriadevoe.com/blog/building-a-id3-decision-tree-classifier-with-python
5.Scikit Learn Implementation of Regression Tree and Random forest
a.https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/tree/_classes.py#L1034
b.https://github.com/scikit-learn/scikit-learn/blob/0d378913b/sklearn/ensemble/_forest.py#L1399

7.5 *ADAPTIVE boosting algorithm*

1.Understanding Adaboost-Understanding Ada Boosting… | by AkashDesarda | Towards Data Science
2.Experiments with AdaBoost.RT, an Improved Boosting Scheme Regression.
(psu.edu)
3.Improving Regressors Using Boosting Techniques.
download (psu.edu)
4.Scikit Learn Implementation of Adaboost Regression Tree Regression
scikit-learn/_weight_boosting.py at
5.Regression Trees
lecture-10.pdf (cmu.edu)

7.6 Support vector regression:

1.Understanding support vector regresssion
(a)https://www.youtube.com/watch?v=ECXc2P6t3TY
2.For loss function
(a)https://alex.smola.org/papers/2004/SmoSch04.pdf
3.Scikit learn implementation of linear kernel
(a)https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
4.Scikit learn implementation of svr kernel
(a)https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html