Machine Learning Engineer Nanodegree

Capstone Project

Satya Raghava Dilip Bhattiprolu
November 10th, 2018

I. Definition

Project Overview

Supervised learning is defined as a mapping function for set of input variables to a given output variable. The goal behind obtaining the mapping function is to predict any output for given set of new input. There are 2 kinds of supervised learning problems:

Classification – A classification problem is to categorize the outputs into set of categories such as is the person “short” or “tall”.

Regression – A regression problem is to predicting actual value of the output such as predicting actual height of person.

New York City is one of the most populous cities in the United States. For an overly crowded place like this, traffic has been a major concern for over several years now. The transportation system here is a network of complex infrastructure that has one of the largest subway systems in the world, an aerial tramway, a vehicular tunnel etc. Taxis account for a significant contribution towards the daily commute of people residing in NYC. There was a survey conducted to know the number of taxi trips per day and it turned out to be a whopping number of around 850,000 trips per day through taxis. This number accounts to approximately 1/10th of the total population in New York City.

This project is a regression problem which predicts the fare amount for a taxi ride in New York City based on set of inputs.

The Dataset is taken from Kaggle https://www.kaggle.com/c/new-york-city-taxi-fare-prediction/data

Problem Statement

This project aims to predict the estimated fare for a taxi trip in New York City for the given pickup and drop off locations. There are several factors that could influence the estimated fare and some of these include the point of pickup, destination, peak hours, high booking requests at the respective location, low availability of drivers, weather, number of passengers, time of booking request etc.

Following are the tasks involved in this project:

- Downloading the train and test dataset from Kaggle.
- Data preprocessing.
- Dividing the input data into training and validation set.
- Training the classifier to predict the taxi trip fare.
- Calculating the Root Mean Square Error for training and validation set.
- Applying the model to predict the fares from test file.

This is a supervised learning regression task as the goal here is to predict the fare for the taxi trip. There are various kinds of regression techniques available to make predictions. These techniques are mostly driven by three metrics (number of independent variables, type of dependent variables and shape of regression line). Some of the Regression models that can be used are linear regression, decision tree regression, random forest regression, polynomial regression, adaboosting, gradient boosting, neural networks. This primarily focus in this project is going to be in the following 3 models

- Linear Regression
- RandomForest Regression
- XGBoosting

Linear Regression is fits a line (or plane) to the data set. This may not work but would help us to see how close we are to benchmark. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target. They are intrinsically suited for multiclass problems. They also work well with a mixture of numerical and categorical features. It's very likely for this to work. XGBoost is a gradient boosting algorithm. The intuition behind gradient boosting algorithm is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better. This model should for most datasets.

Metrics

Mean Absolute Error (MAE) and Root mean squared error (RMSE) are two of the most common metrics used to measure accuracy. Both MAE and RMSE express average model prediction error in units of the variable of interest. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable.

Since we don’t want large errors we use RMSE as the metric to choose the model.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

II. Analysis

Data Exploration

Following are the input features and target variable from the given test data set:

Features:

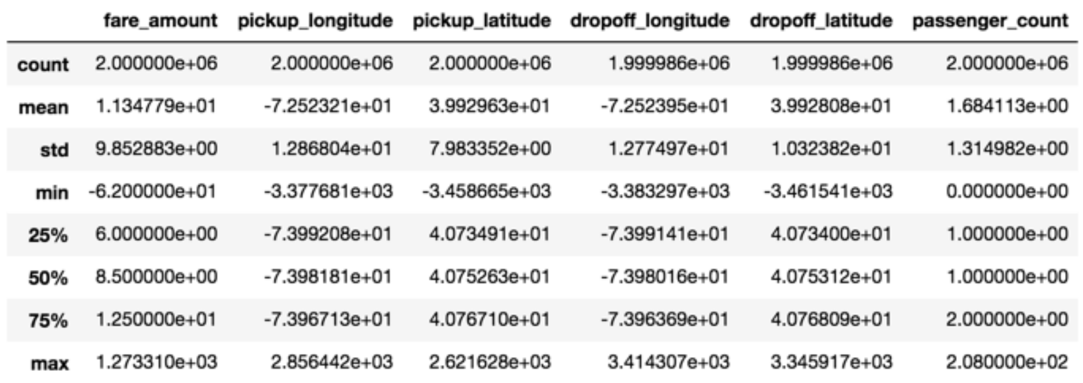
- pickup_datetime - timestamp value indicating when the taxi ride started.
- pickup_longitude - float for longitude coordinate of where the taxi ride started.
- pickup_latitude - float for latitude coordinate of where the taxi ride started.
- dropoff_longitude - float for longitude coordinate of where the taxi ride ended.
- dropoff_latitude - float for latitude coordinate of where the taxi ride ended.
- passenger_count - integer indicating the number of passengers in the taxi ride.

Target:

- fare_amount - float dollar amount of the cost of taxi ride. This value is only in the training set; this is what we are predicting in the test set and it is required in our submission CSV.

The input training data set contains 55 Million rows. Two million rows are used in the training model.

Here is how initial input data looks before preprocessing.



	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
count	2.000000e+06	2.000000e+06	2.000000e+06	1.999986e+06	1.999986e+06	2.000000e+06
mean	1.134779e+01	-7.252321e+01	3.992963e+01	-7.252395e+01	3.992808e+01	1.684113e+00
std	9.852883e+00	1.286804e+01	7.983352e+00	1.277497e+01	1.032382e+01	1.314982e+00
min	-6.200000e+01	-3.377681e+03	-3.458665e+03	-3.383297e+03	-3.461541e+03	0.000000e+00
25%	6.000000e+00	-7.399208e+01	4.073491e+01	-7.399141e+01	4.073400e+01	1.000000e+00
50%	8.500000e+00	-7.398181e+01	4.075263e+01	-7.398016e+01	4.075312e+01	1.000000e+00
75%	1.250000e+01	-7.396713e+01	4.076710e+01	-7.396369e+01	4.076809e+01	2.000000e+00
max	1.273310e+03	2.856442e+03	2.621628e+03	3.414307e+03	3.345917e+03	2.080000e+02

The table shows some obvious outliers. For example, the max passenger count is 208. The mean mean fare_amount is 11.3. Therefore max fare_amount 1273.3 looks like an outlier. The data cleaning is explained further in data preprocessing. Distance is calculated based on pickup and dropoff latitude and longitude.

Exploratory Visualization

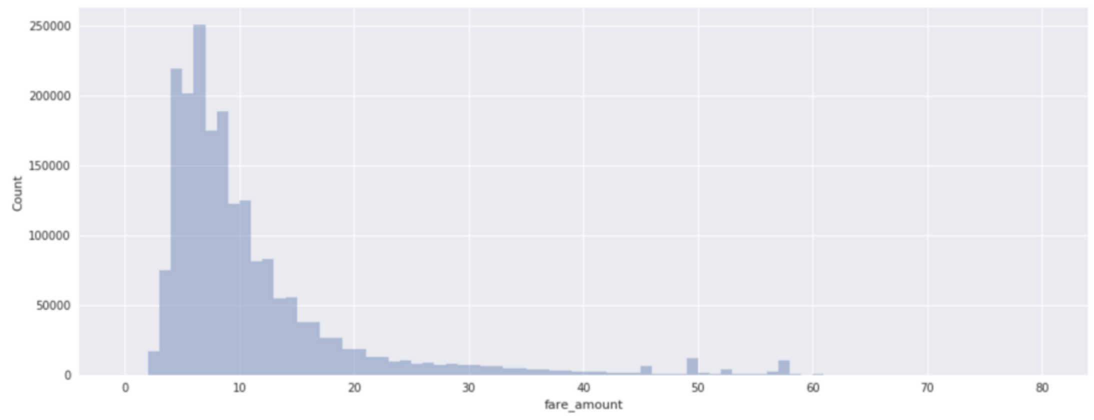


Figure1

Figure1 depicts the distribution of fare_amount for 2 million rows. Most of the fare_amount is under 10$.

Certainly, the fare_amount depends on distance. The input features consists of pickup latitude, pickup longitude and drop off latitude and dropoff longitude. To obtain the approximate distance based on the latitude and longitudes, geopy library was used. Figure2 depicts the distribution by distance.

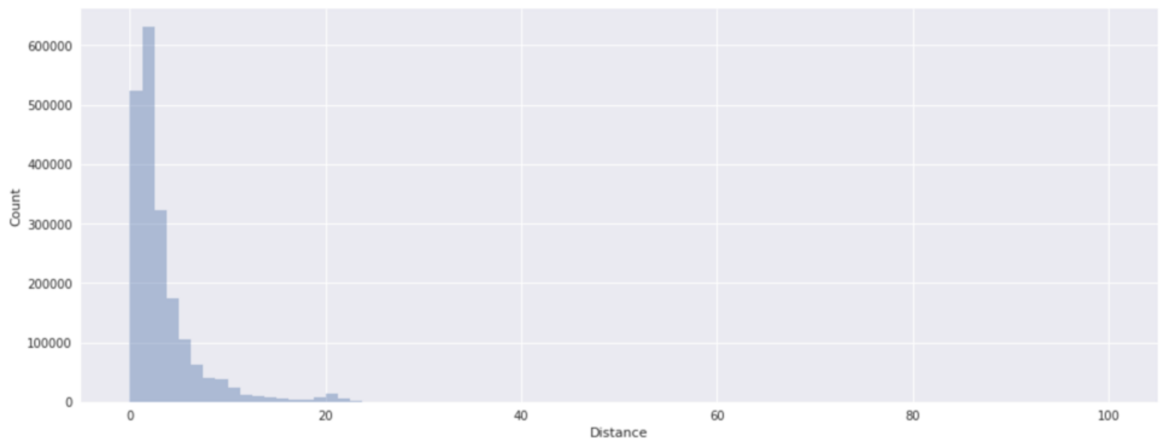


Figure2

It is evident from Figure2 that most of the input training set have a distance under 10 miles.

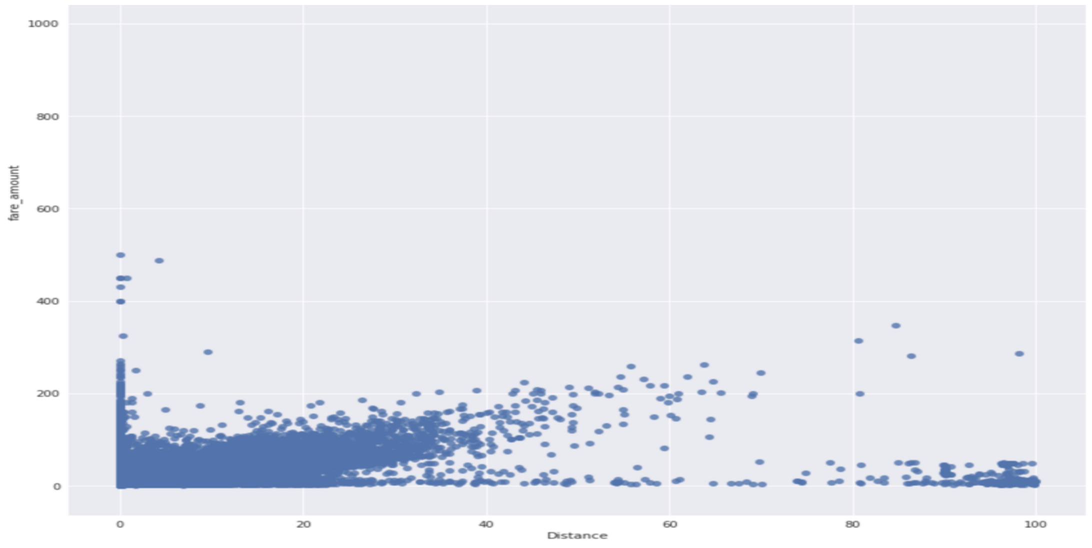


Figure3

Figure3 is the scatter plot distribution between fare_amount and distance.

Fare_amount also depends on the date and time of pickup for a given day. The input timestamp is divided into day, month, year, hour and day of the week.

Box plots are used to observe the variation in fare_amount with respect to day, month, hour, and day of the week.

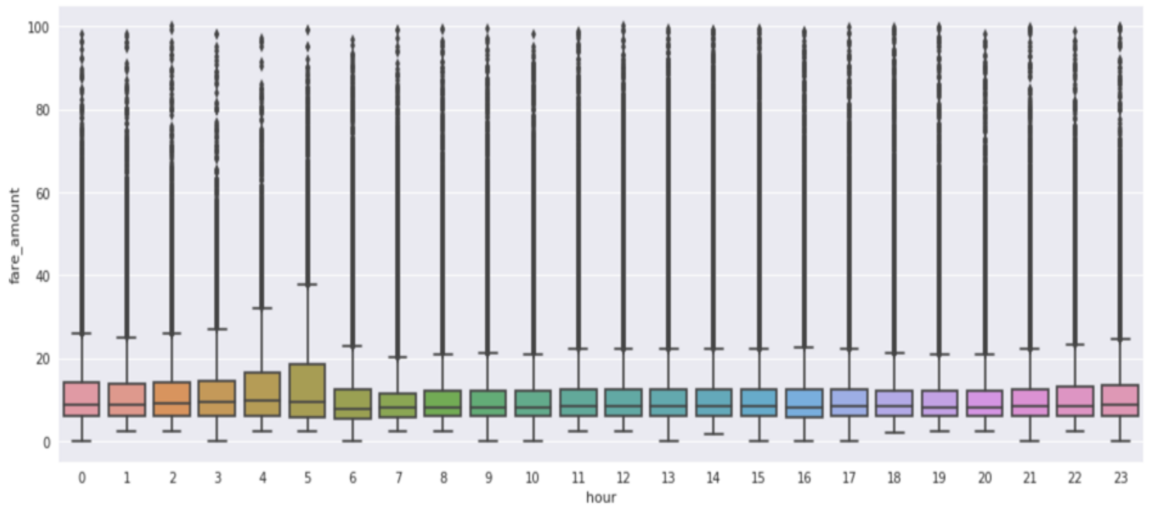


Figure4

Figure4 shows distribution of fare_amount which is higher during early hours and late hours of a day. This could be due to the limited availability of taxis at the time.

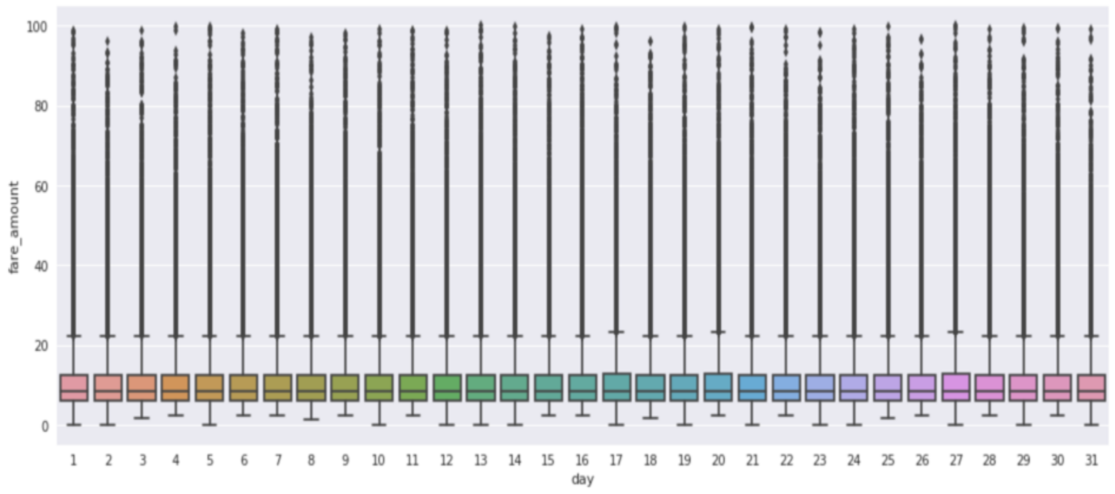


Figure5

Figure5 shows fare_amount distribution which looks similar in most of the days in a given month.

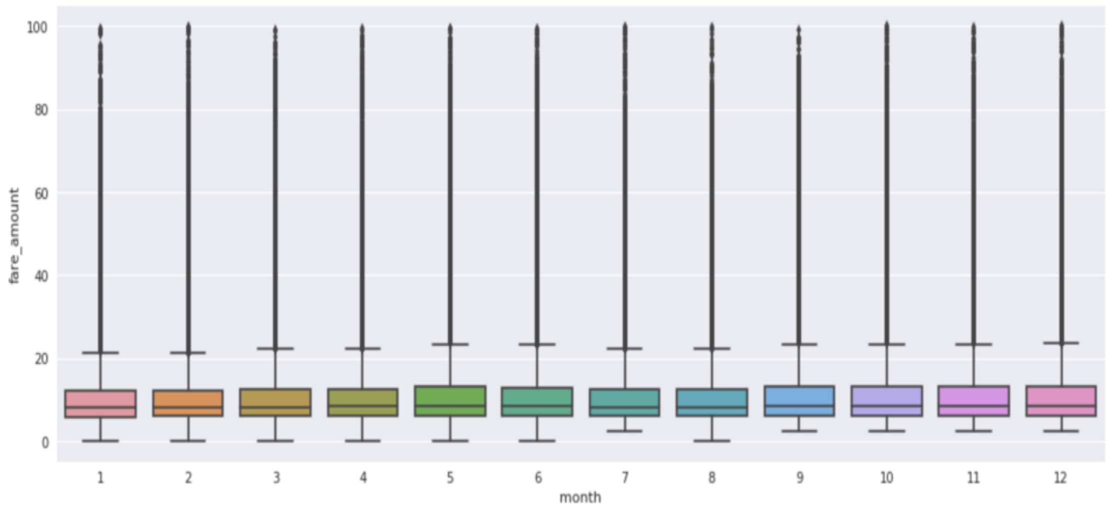


Figure6

Figure6 shows fare_amount distribution which looks higher from months 3-12.

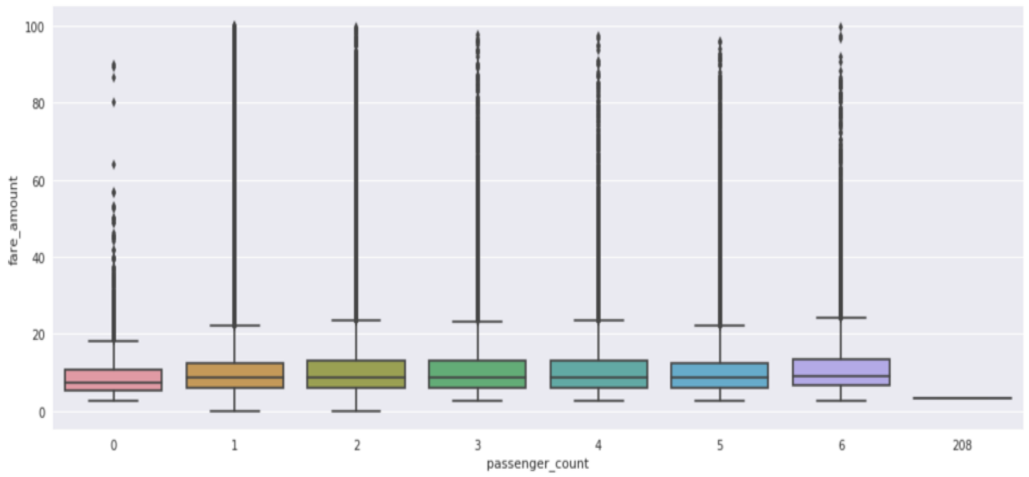


Figure7

Figure7 shows fare_amount distribution for passengers 1 to 6 which resembles the same.

Data assumptions are discussed in the Data Preprocessing section.

ALGORITHMS AND TECHNIQUES

This clearly is a supervised machine learning regression task. To resolve this problem the following models are used:

- Linear Regression
- Random Forest Regression
- XGBoost

Linear Regression: Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, the method is known as multiple linear regression.

- Simple Linear Regression: The below is the equation for simple linear equation.

  $$y(x) = w0 + w1 * x$$

  where w's are the parameters of the model, x is the input, and y is the target variable. Different values of w0 and w1 will give us different lines.

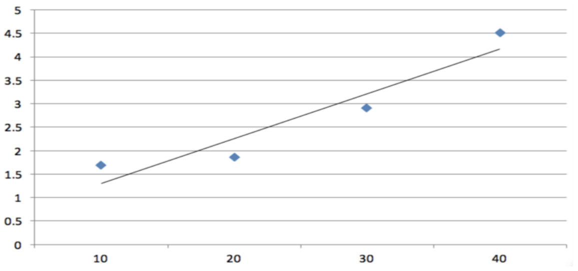


Figure8

- The case when we have more than one feature is known as multiple linear regression, or simply, linear regression. We can generalize our previous equation for simple linear regression to multiple linear regression:

  $$y(x) = w0 + w1 * x1 + w2 * x2 + \ldots\ldots + wn*xn$$

  In the case of multiple linear regression, instead of our prediction being a line in 2-dimensional space, it is a hyperplane in n-dimensional space. For example, in 3D, our plot would look as follows

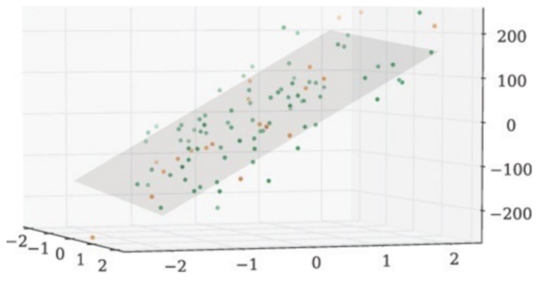


Figure9

Pros: It’s simple and captures linear relationships
Con: It’s not very useful for practical applications as most real world relationships are non-linear.

RandomForest Regression: . Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of “weak learners” can come together to form a “strong learner”. Each classifier, individually, is a “weak learner,” while all the classifiers taken together are a “strong learner”. Random Forests are simply an ensemble of decision trees. The input vector is run through multiple decision trees. For regression, the output value of all the trees is averaged.

$$g(x)=f0(x)+f1(x)+f2(x)+...$$

where the final model g is the sum of simple base models fi. Here, each base classifier is a simple decision tree.

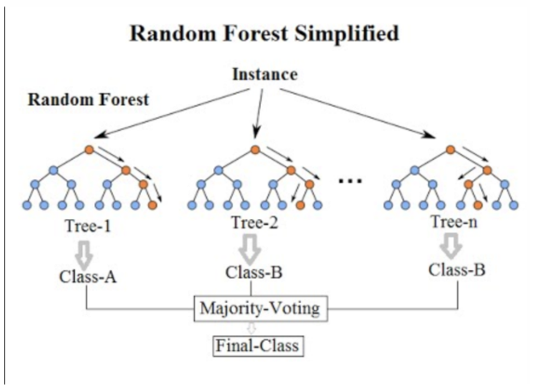


Figure10

Pros:

- Random forest algorithm can be used for both classifications and regression task.
- Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data.
- If there are more trees, it won’t allow overfitting trees in the model.

Cons:

- The main drawback of Random Forests is the model size. You could easily end up with a forest that takes hundreds of megabytes of memory and is slow to evaluate.
- Another point that some might find a concern is that random forest models are black boxes that are very hard to interpret.

XGBoost:

This is an ensemble method that seeks to create a strong classifier based on weak classifiers. In this context, weak and strong refer to a measure of how correlated are the learners to the actual target variable. By adding models on top of each other iteratively, the errors of the previous model are corrected by the next predictor, until the training data is accurately predicted or reproduced by the model
Now, gradient boosting also comprises an ensemble method that sequentially adds predictors and corrects previous models. However, instead of assigning different weights to the classifiers after every iteration, this method fits the new model to new residuals of the previous prediction and then minimizes the loss when adding the latest prediction. So, in the end, you are updating your model using gradient descent and hence the name, gradient boosting. This is supported for both regression and classification problems. XGBoost specifically, implements this algorithm for decision tree boosting with an additional custom regularization term in the objective function.

Pros:

- Execution speed and Model Performance.

Benchmark

The author of this dataset in Kaggle calculated the basic estimate based on merely the distance between two points which resulted in an RMSE of \$5-\$8. Linear Regression has given a Score of \$6. The project utilizes \$5 as a benchmark to make a decision on the model.

III. Methodology

Data Preprocessing

The primary goal of data cleaning is to detect and remove errors as well as anomalies. Following are the steps used for data preprocessing:

- Few rows contained fare_amount < 0. These rows have been removed.
- Removed the rows containing cells with null values.
- Latitude range is -90 to 90 and Longitude range is -180 to 180. Rows falling out of range have been removed
- Distance is calculated using geopy library based on pickup latitude, pickup longitude, dropoff latitude, and dropoff longitude.
- Distance travelled in taxis is usually under 300 miles. There are around 4k records which are over that. Clearly, this is because of an incorrect data entry. These are numerous records to be dropped. An attempt was made to correct the approximate distance for these values based on the fare_amount. Following assumptions were made based on an online search.

    - Minimum fare amount in nyc = 2.5
    - Cost for every 1/5 mile = 40
    - Cost for 1km = 40*5/1.6 = 125 cents = 1.25$
    - fare_amount = 2.5 + (dist * cost for 1km)
    - dist = (fare_amount - 2.5)/1.25

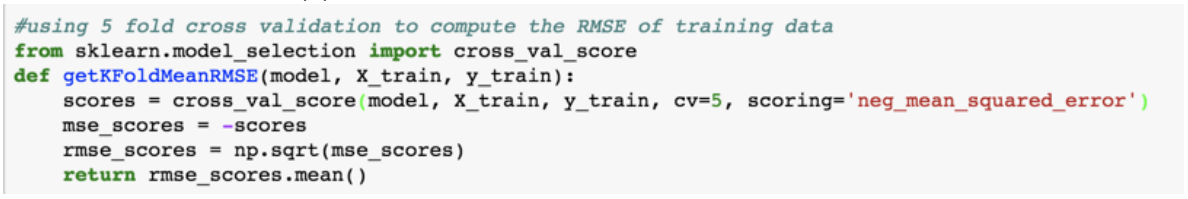
- Passenger count varies from 0 to 208. In general a maximum of 6 passengers can be accommodated in a big taxi. Considering the count exceeding 6 as an outlier, all the values beyond 6 were dropped. There were a few with count of 0. Online search showed that these could be charges for those who booked a taxi and cancelled them last min.

- The pickup date time is converted into day, month, year, hour, day of week. The pickup_datetime was dropped from the input data set.
- Key has been dropped since it does not provide any relevant information that can help in the prediction.

IMPLEMENTATION

Kfold cross validation technique has been used. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k − 1 folds. cross_val_score from sklearns model selection has been used to calculate the score.

Here is the code snippet:

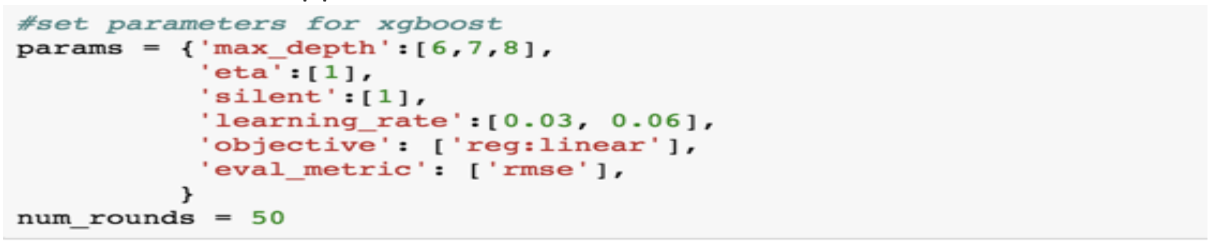


```
#using 5 fold cross validation to compute the RMSE of training data
from sklearn.model_selection import cross_val_score
def getKFoldMeanRMSE(model, X_train, y_train):
    scores = cross_val_score(model, X_train, y_train, cv=5, scoring='neg_mean_squared_error')
    mse_scores = -scores
    rmse_scores = np.sqrt(mse_scores)
    return rmse_scores.mean()
```

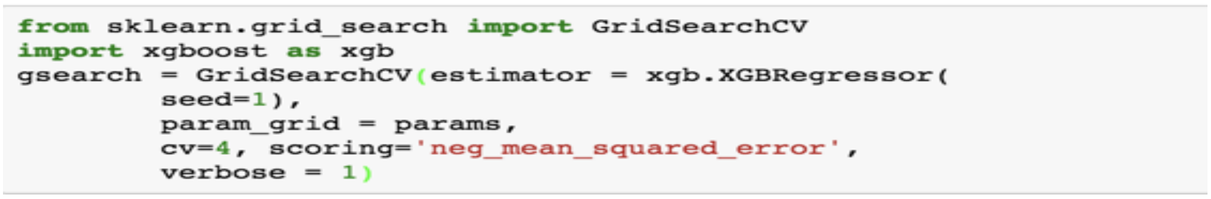
The Linear and RandomForest regression with default parameter set have been applied on the dataset to get the RMSE using the above defined function.

On the otherhand, used GridsearchCV to tune the parameters of XGBoost and get the best param set for predicting the model.

Here is the code snippet:



```
#set parameters for xgboost
params = {'max_depth':[6,7,8],
          'eta':[1],
          'silent':[1],
          'learning_rate':[0.03, 0.06],
          'objective': ['reg:linear'],
          'eval_metric': ['rmse'],
         }
num_rounds = 50
```



```
from sklearn.grid_search import GridSearchCV
import xgboost as xgb
gsearch = GridSearchCV(estimator = xgb.XGBRegressor(
        seed=1),
        param_grid = params,
        cv=4, scoring='neg_mean_squared_error',
        verbose = 1)
```

LinearRegression was the fastest among the three. RandomForest regressor took around 20mins. XGBoosting used hyperparameter tuning which means the algorithm was run for k iterations(for k fold) for every combination of parameters. In total 3*2= 6 hyperparameters were used and 4 fold cv has been applied. So there were 6*4=24 iterations were applied and the best parameters from those were obtained. This algorithm took over an hour time to get the results.

IV. RESULTS

MODEL EVALUATION AND VALIDATION

Cross Validation is a very useful technique for assessing the performance of machine learning models. It helps in knowing how the machine learning model would generalize to an independent data set. K-Fold Cross Validation is a common type of cross validation that is widely used. In every iteration, root mean square values for the algorithm are computed. At the end of all iterations the mean score is computed and is used as overall score for the algorithm. Therefore the scores are more accurate.

RMSE scores obtained by using the following models are as follows:

- Linear Regression: This method tries to fit a simple line/plane for the input data set. The scores obtained by using this model are as follows:
    - 5 fold CV RMSE score: 6.67

  Clearly this is above the benchmark of RMSE $5 and therefore it is not the correct model.
- RandomForest Regression: The Random Forest is an ensemble of Decision trees which produces great results even without hyper-parameter tuning. It is a bagging algorithm that reduces variance. The following score are obtained using this algorithm:
    - 5 fold CV RMSE score: 4.53

  The score meets the benchmark.
- XGBoost: XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. GridSearchCV has been used to obtain the best suiting parameters for the data. Score obtained using this model is
    - 3 fold RMSE score using GridSerachCV: 4.29

Clearly the benchmark has been met. The score has improved compared to Random Forest. Tuning parameters further could bring even more improvement

Justification

The proposed benchmark was achieved by trying RandomForest and XGBoost that were among the initial guesses to predict the model. The RMSE for both the models have achieved the benchmark. Therefore we can go with one of those models. However I lean towards using XGBoost as it is easier to fine tune the model.

V. Conclusion

Linear Regression doesn't meet the given benchmark. RandomForest Regression and XGBoost meet the benchmark. RandomForest performs best on the training data. However, it could be overfit the training data. XGBoosting is fast and it performs well on training and testing data. It is possible to fine tune the hyper parameters and enhance the training, testing scores for XGBoosting using Grid Search.

The following is the feature importance obtained from XGBoost. As expected distance was the most critical feature in determining the fare-amount.

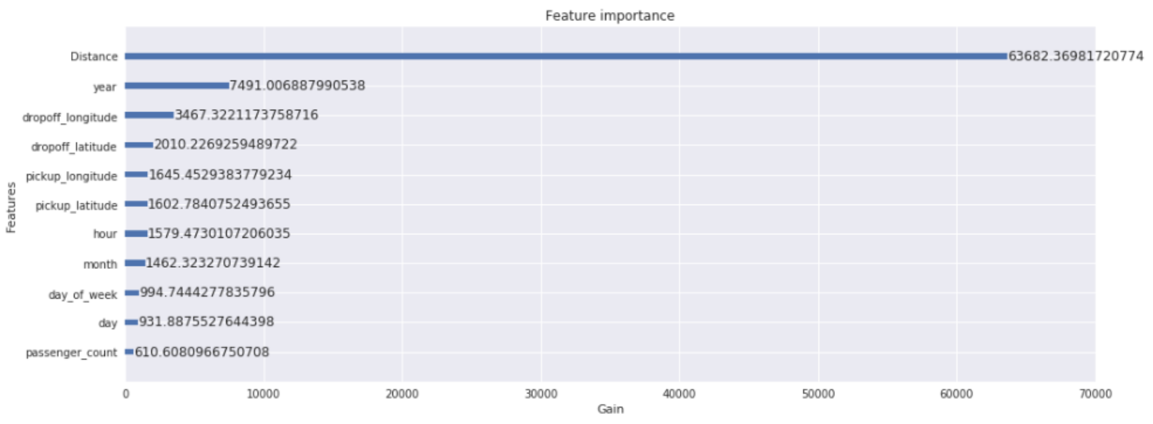


Figure11

Reflection

The following are the high level steps involved:

- Data Preprocessing.
- Model Selection.
- Predicting Values for the test set using the model.

Crucial piece for the project was data cleaning and preprocessing. This was the first time where I did extensive data preprocessing. Visualizations were helpful in understanding the data set. They also helped in preprocessing of data.

Linear Regression and RandomForest Regression were applied as mentioned in Proposal. Reviewer recommended to try XGBoost. This is the first time I used XGBoost and found it very useful in training supervised learning models in general. Initial approach was to use train test split from sklearn and compute the training and testing scores. This approach was fast as it computed the train test score only once. After the initial review the approach has been changed to use the k-fold CV. The biggest challenge in doing this was to wait for several minutes. Initially I had doubts were if the system hanged in between execution. After multiple iterations and waiting long enough results were obtained. Overall it took at least 2 hrs for the entire notebook to run and fetch the results.

Improvement

Out of 3 approaches, I like XGBoost as there is scope for improvement. I only played around with the max_depth(6,7,8) and the learning rates(0.03 and 0.06) as they influence the most in the model. We could try more options for these two parameter or could try tuning other parameters. Other non linear approaches can also work better than the approaches used. Deeplearning can also be applied and a neural network can be build to solve this regression problem.

References

- https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/
- https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d
- https://elitedatascience.com/python-seaborn-tutorial
- http://nymag.com/nymetro/urban/features/taxi/n_20286/
- https://medium.com/datadriveninvestor/understanding-linear-regression-in-machine-learning-643f577eba84
- https://www.kdnuggets.com/2017/10/random-forests-explained.html