CS 634 Project
New York City Taxi Trip Duration Prediction

Kavan Patel Keyur Patel

I. Introduction

This is a Kaggle Competition where participants have to predict a trip duration of a given taxi ride. Goal is to improve prediction of trip duration time in order to improve electronic dispatching system. Accurate prediction of trip duration will help dispatcher to know how long will it take cab driver to reach the destination and after what time cab will be available for the next ride and this will help scheduling next ride for all the cab. In this report we propose three algorithmic solutions to approach this problem, that can achieve RMSLE score less than 0.5. Note that 0.28976 is the best score by the winning group among all the competitors.

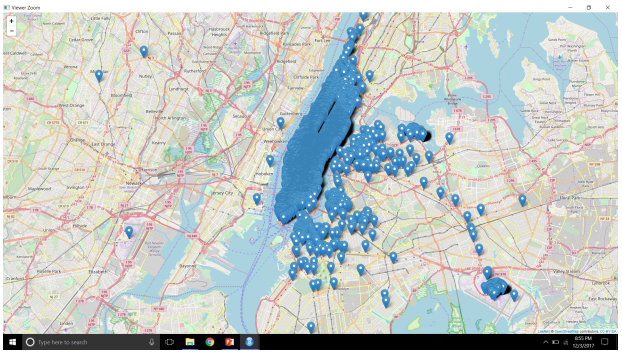

Fig. 1. Trend of data

II. Data set

The data set is released by NYC Taxi and Limousine Commission, it consists of attributes like pickup time, drop off time,Geo-coordinates, number of passengers,Vendor id. Most of the data belongs to Manhattan area of NYC.
Number of tuples : 1.4M in training data, 625k in test data. Evaluation Problem for this problem is Root Mean Square Log Error (RMSLE).

III. Prepossessing and Cleaning

Cleaning: We do not have missing value of in our data set but we had some skeptical value which can effect the model while training the model.
Removing Outlier: The traditional method of removing outlier is a Boxplot so we use that method in order to remove the outlier. IQR of Boxplot is 678 and the first quantile and the third quantile are respectively 397 and 1075, and so trip duration more than 1.5 * 678 + 1075 = 2767 will be considered as outleir and which is not a practical approach to remove the outleir because it will remove more 30 percent of the data.The reason we are getting such results is because the distribution of the data is not a normal distribution as we can see in the histogram. We choose to apply histogram to see the distribution of the data, after looking at the distribution f the data we decided to remove the all the tuples which has higher trip duration than 12000 and lower trip duration than 120.

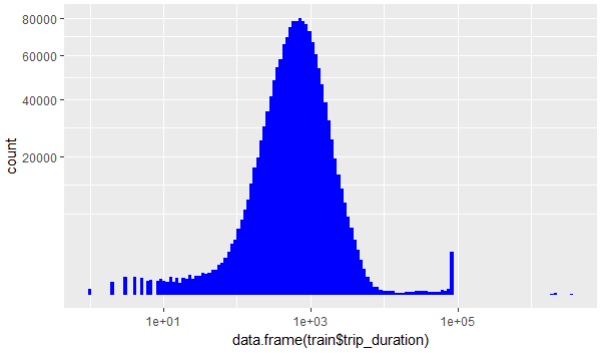

Fig. 2. Trend of data

IV. Algorithms

The dataset contains geo-spatial longitude and latitude co-ordinates. The problem with this type of data is that, the difference in values between any two points (any two co-ordinates) will be very small. Eg: Distance between points (-73.98812866, 40.73202896) and (-73.96420288, 40.67999268) is around 3 km, but the these values appear to be very close to each other as the have very small difference. So the algorithm will take both values as same and it will end up losing some information about the pickup point or dropoff point being different from eachother. So we need to transform these values in such a way that the fact that each point is different from eachother is prevailed and also that each pair of value belongs to a 2-dimensional plane.
In order to tackle this problem, we thought to transform the data with Principle Component Analysis (PCA). Ideally PCA is used to reduce number of dimensions, but the beauty of PCA is it transforms the data in a new co-ordinate system, which is exactly what we want! It transforms each pair of values in a new co-ordinate system, thereby retaining the information of each point being a unique point yet significantly different from other pairs of latitude longitude co-ordinates.
We took 2 cases, where the data is PCA transformed and

another as a baseline where each co-ordinates are transformed to its ceiling value.

A. Random Forest

Random Forest regressor is an ensemble of decision trees which can also be thought as "weak estimators". It makes predictions by combining different base models (trees) where each model is constructed independently by using different subsample of input data. One thing to be noted is that this is a regression model which works differently than Random Forest Classifier. Instead of splitting branches of the tree by entropy, this model uses Sum of Squared Error (SSE) as a parameter to split the branches, as the predicted value is a continuous attribute. Attribute with lowest SSE value is chosen as a split.

1) Hyper-parameter tuning: There are number of hyper-parameters that are to be fine-tuned to produce good predictions. We tuned 2 hyper-parameters: *ntrees* and *max_depth*. *ntrees* sets the number of trees that are to be ensemble. Normally, the performance increases as the increase in number of trees. So the number of trees should be chosen in a way that predictions are good enough and should also be compatible with processor as it will slow down the performance. 1000 trees were chosen as value of *ntree* parameter. *Max_depth* parameter decides the depth of trees or number of nodes in a tree. Its value was decided by experimentation. Figure 3 shows the performance of Random Forest regressor on various values of *max_depth* attribute.

B. Neural Network

Neural Networks are one of the best and hottest algorithms amongst data science community. It is a connection of interconnected layer of nodes (neurons) where each node provides some unique information. It consists mainly 3 layers, input, output and hidden. Deep Neural Networks however have more than 2 hidden layers. We used a network with 1 hidden layer consisting of 23 neurons, for the prediction of trip duration.

1) Hyper-parameter tuning: There are various hyper-parameters to be tuned for a good performance of neural network. Few of them are: Number of *hidden layers*, *hidden layer size*, *learning rate*, *activation function*. We thought it is out of scope of this project to use deep neural networks with hidden layers more than 2. So we limited the number of layers to 1. Hidden layers size is the number of neurons in a hidden layer. Increasing number of neurons in hidden layer may increase the prediction performance but it will reduce after it exceeds a particular number. Learning rate is a parameter to speed up the learning process but it should be chosen carefully as it may end up overshooting the global minima. Activation function is a nonlinear function used to map the non-linearity of the data. We used ReLU as activation function. Beauty of ReLU is that if its input value is negative than it won't activate the neuron. This means that at a time only a few neurons are activated making the network sparse making it efficient and easy for computation. All the values of hyper-parameters were decided by experimentation. Figure 4 shows the performance of Neural Network regressor with various values of hyper-parameter.

C. Gradient Boosting Machine

Gradient Boosting Machine is an ensemble of predictors (trees), where predictors are added sequentially to an ensemble each one correcting its predecessor. This method tries to fit the new predictor to the residual errors made by the previous predictor. For example, lets go through a simple regression problem with decision trees as base predictors.
First tree fits the training set. Lets call *e1* as error associated with this tree. Now second tree is trained on this residual error *e1* and generates error *e2* associated with training set. Next, third tree is used to fit the residual error *e2* with training set giving residual error *e3*. Now we have ensemble of three trees, it can make predictions simply by adding the predictions of all trees. However its arbitrary to choose the number of trees but it can be chosen by experimentation. One of the important hyper parameter is learning rate. This parameter scales the contribution of each tree. If it is very small like 0.01, then we need a large number of trees to fit the model.

1) Hyper-Parameter tuning: We chose to tune 3 hyper-paramteres : *ntrees*, *max_depth* and *learning-rate*. *ntree* was set to 1000, it was an arbitrary choice to be compatible with processing power of the machine. *learning-rate* was set to 0.01 as we have a large number of trees. *max_depth* was tuned by experimentation. Figure 5 shows the performance of GBM with various *max_depth* values.

V. Results

We performed prediction tests on two datasets, one with latitude and longitude transformed in XY plane by PCA and the other where latitude and longitude converted to its ceiling as a baseline performance.

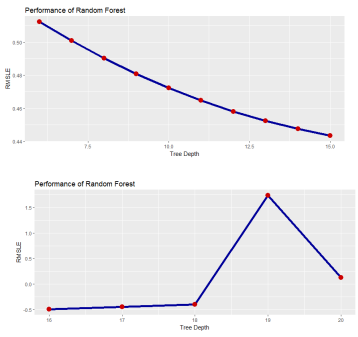

Fig. 3. Performance of Random Forest on PCA transformed data

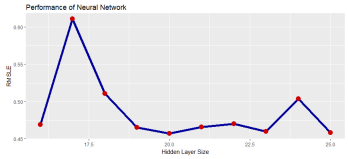

Fig. 4. Performance of Neural Network on PCA transformed data

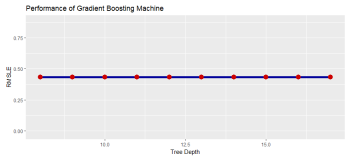

Fig. 5. Performance of Gradient Boosting Machine on PCA transformed data

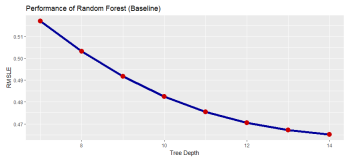

Fig. 6. Performance of Random Forest on Baseline data

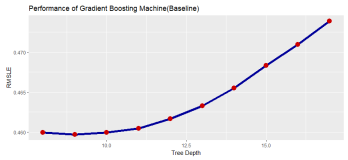

Fig. 7. Performance of Gradient Boosting Machine on Baseline data

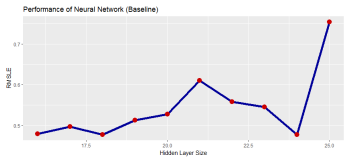

Fig. 8. Performance of Neural Network on Baseline data

VI. Conclusion

After using three algorithms, we can conclude that Gradient Boosting Machine outperformed the other two, yet there was no significant difference on the performance. Best RMSLE value for GBM is 0.4496, 0.47 for Neural Network and 0.445 for Random Forest over PCA transformed data.

While on the Baseline data, GBM performed best with 0.46 RMSLE score, 0.47 for Neural Network and 0.465 for Random Forest.

One more thing to conclude, though we tried to maintain the information for each point being a unique pickup/dropoff point on a plane, there was no significant difference in the performance of the algorithm as compared to the performance over baseline data, pretty much surprising that approximation of such a huge data yet performed as close as the original data.

We also did the performance test on test data, and uploaded on Kaggle.com to get the actual RMSLE score to see where we stand among other competitors. We managed to score 0.4495 RMSLE on the GBM model, where as 0.28976 by the winner of the competition.

References

[1] https://www.kaggle.com/c/nyc-taxi-trip-duration/kernels
[2] Hands-On Machine Learning with Scikit-Learn and Tensorflow
[3] https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/
[4] http://setosa.io/ev/principal-component-analysis/