Taxi Fare Prediction

Hongye Xu
Rutgers University
hx123@scarletmail.rutgers.edu

Shengjie Li
Rutgers University
shengjie.li@rutgers.edu

Abstract

The goal of our project is about applying data processing skills and machine learning algorithms to predict New York City taxi fares base on the dates, times and coordinates data. The feature engineering focus on figure out the relationship between geographical features and fare amount. Finally we implement 3 most popular decision tree algorithms and neural network architecture on the dataset and compare their performances.

1 Introduction

The cost of a taxi ride is related to many different things. Such as the distance of the travel, the time of the travel, even the route of the travel (toll roads/toll bridges). Given this complicated nature of the pricing of taxi rides, our project is mainly about estimating the price of taxi rides in NYC using the following features:

- Date and time of the taxi ride.
- The longitude and latitude coordinate of where the ride started and ended.
- Number of passengers.

Applying machine learning methods in real life situations is always facinating. Predicting the cost of taxi rides is a real life task and its easy to apply different machine leanring methods on this task. We have gone through many discussions about supervised regression tasks and we have seen many methods like RNN, XGboost, Ramdomforest, etc. Doing this project would give us some experience on developing machine learning methods on real life tasks.

In this report, we will first introduce the dataset we use. Then we will list the experiments we have done. Finally, we will conclude the results and demonstrate the future works

2 Dataset

We use the dataset from a kaggle competition *New York City Taxi Fare Prediction – Can you predict a rider's taxi fare?*[1]. The dataset is provided by Google Cloud. It contains more than 55 Million rows in the dataset. The first 5 rows are shown in Table 1. Each row in the dataset contains the following fields:

- **key**: A string, generated by adding a unique integer to the pickup_datatime, and served as a unique id to identify different rows.
- **fare_amount**: A floating point number, the total cost of the taxi ride.
- **pickup_datetime**: A timestamp, the exact date and time of the pickup.
- **pickup_longitude**: A floating point number, longitude of the pickup location.
- **pickup_latitude**: A floating point number, latitude of the pickup location.
- **dropoff_longitude**: A floating point number, longitude of the drop-off location.
- **dropoff_latitude**: A floating point number, latitude of the drop-off location.
- **passenger_count**: An integer , indicating the number of passengers in the taxi ride.

2.1 Preprocessing

We use Spark as our tool for data preprocessing. We remove some invalid data points such as points with null values, negative fare amount, too many people in a taxi, or longitude and latitude more than

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

key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
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
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
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
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
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

Table 1: The first 5 rows in the dataset

90. The **pickup_datetime** attribute in the dataset is UTC time. As the time might be a crucial point for the taxi cost, we convert it to New York local time – Eastern time.

2.2 Feature Engineering

2.2.1 Geographical features

The coordinate of the center of New York City is (40,-74). However, we found that some pickup and drop-off locations are very far from this coordinate, which indicates a lot of outliers in the dataset. In order to achieve better prediction results, we exclude outliers according to the latitude and longitude range of the test dataset.

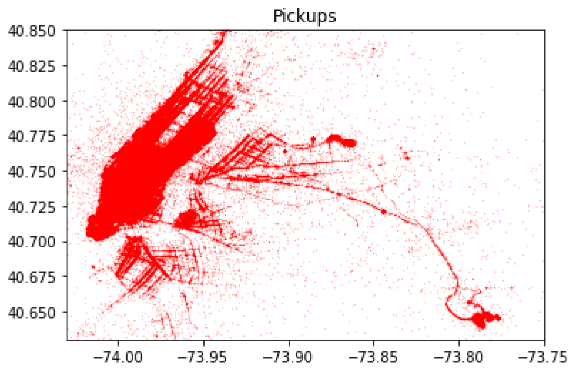

Figure 1: Pickups Heatmap

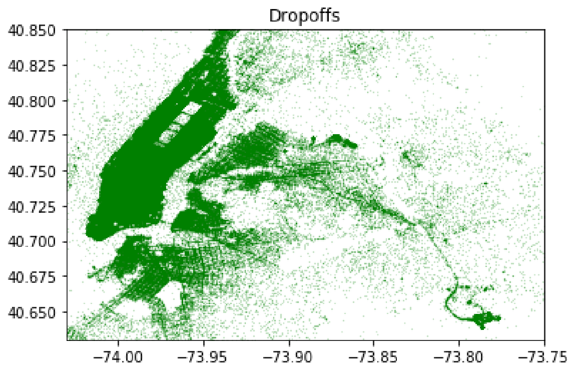

Figure 2: Drop-offs Heatmap

From scatter plot Figure 1 and 2 we can observe the high density of pickups and drop-offs from and to JFK and La Guardia Airport. According to experience, taxis to the airport are generally more expensive, and we take JFK as an example to verify it.

In the training dataset, there are 14961 trips with pickup from JFK and 7686 trips with drop-off to JFK.

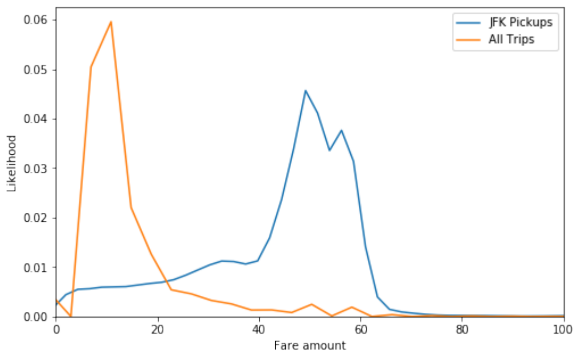

Figure 3: Fare distribution for pickups from JFK and other places

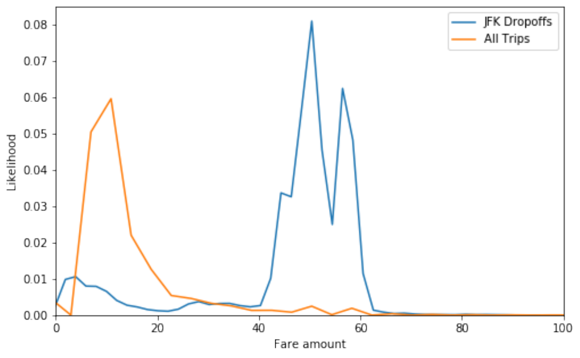

Figure 4: Fare distribution for drop-offs to JFK and other places

As shown in Figure 3 and 4, the distribution of fare amount for both pickup and dropoff to JFK are similar. Obviously, trips starting and ending in JFK are significantly more expensive, so we believe that landmarks will obviously affect the fare of a ride. So we decided to first calculate haversine distances between the pickup and drop-off locations based on the coordinates. (Fare amount and distance are strongly correlated as shown in Figure 5) Then we calculate the distance of pickup locations and drop-off locations from the landmarks in New York, and finally add these two distance together as the aggregate distance from the landmarks.

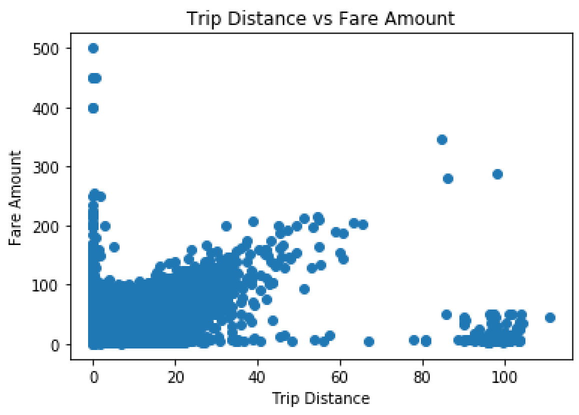

Figure 5: Trip Distance vs Fare Amount

The list of landmarks we choose is as follows:

- **JFK**: John F. Kennedy International Airport.
- **EWR**: Newark Liberty International Airport.
- **LGA**: LaGuardia Airport.
- **SOL**: Statue of Liberty.
- **NYC**: The Central of New York City.

The coordinates of these landmarks are shown in Table 2.

2.2.2 Chronological features

The date and time of the pickup might be key information related to the cost of taxi rides. Thus, We extract hour, day of week, day, month, year from the **pickup_datetime** attribute in the dataset.

The distribution of day of week and months are shown in Figure 6 and 7

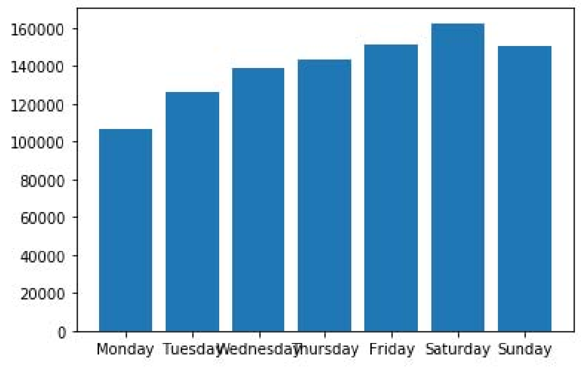

Figure 6: Distribution of day of week

3 Experiments

We train and test multiple models on our dataset to predict taxi fare.

3.1 Models

LightGBM LightGBM is an open-source framework for gradient boosted machines. LightGBM propose two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). GOSS can obtain quite accurate estimation of the information gain with a much smaller data size. With EFB, we bundle mutually exclusive features,to reduce the number of features. The optimal bundling of exclusive features is NP-hard, but a greedy algorithm can achieve quite good approximation ratio (and thus can effectively reduce the number of features without hurting the accuracy of split point determination by much)(Ke et al., 2017). We use a learning rate of 0.05 and 31 leaves for our training.

Xgboost The Xgboost is a scalable and accurate implementation of gradient boosting machines and it has been proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. It is a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning and achieve state-of-the-art results on many machine learning challenges(Chen and Guestrin, 2016).

Random Forests The random forests is a classification algorithm consisting of many decisions trees(Breiman, 2001). It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. The model is trained for 100 iterations with 3-fold cross validation.

MLP We also use a Multi-layer Perceptron as our Neural Networks representative. Our MLP model contains 5 hidden layers of size 256, 128, 64, 32, 8, respectively. Every hidden layer is followed by a ReLU activation function and a BatchNorm layer. Adding BatchNorm layers accelerates the training process(Ioffe and Szegedy, 2015). This model is trained with Adam optimizer with a learning rate of 0.001.

	JFK	EWR	LGA	SOL	NYC
latitude	40.639722	40.6925	40.77725	40.6892	40.7141667
longitude	-73.778889	-74.168611	-73.872611	-74.0445	-74.0063889

Table 2: Landmarks Coordinates

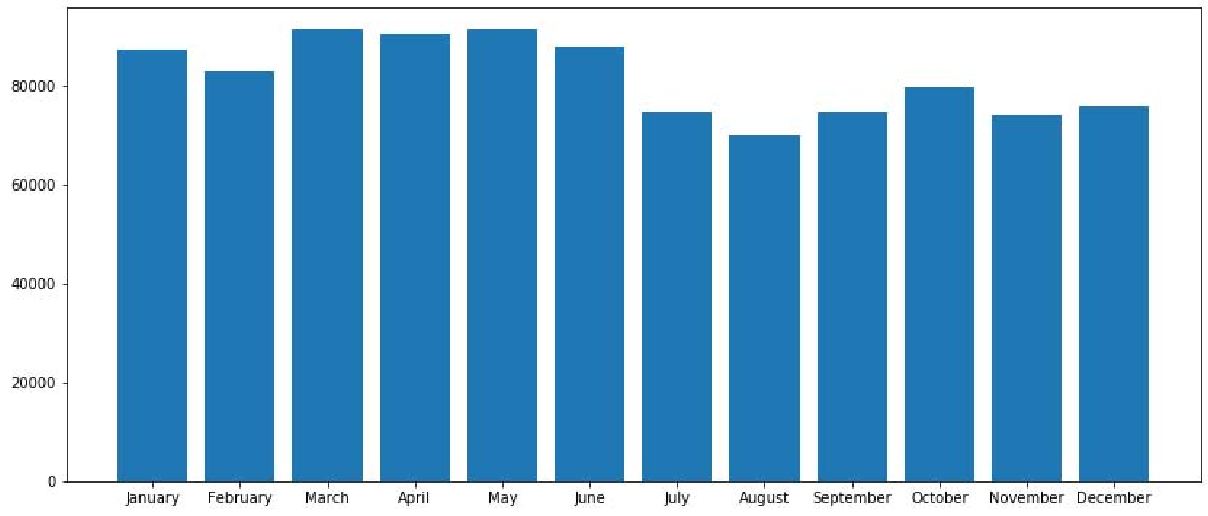

Figure 7: Distribution of months

3.2 Results

The results of all of our models are presented in table 3, where we present the mean absolute error, the root mean square error, as well as the Kaggle score. The Kaggle scores are obtained by submitting prediction files to the Kaggle competition. For the interested reader, we present in the appendix A the top features of different models. The LightGBM model achieves best performance on all three metrics.

4 Conclusion

The goal of our project is about applying data processing skills and machine learning skills to real life problems. Predicting taxi fare is an ideal problem for these purposes. To solve this problem, we first need to obtain the data, clean and preprocess the data, extract useful information. During this process, we are able to utilize all the different data processing tools we learned such as Spark. After we have the data, we need to build Machine Learning models to make predictions, which allows us to implement all the different models we have learned from class. In this project, we implemented many models, from gradient boosting decision trees to neural networks. Our understandings to these models are further improved during the implementation process.

Apart from the models in the results rable 3, we have also tried other models such as Support Vector Regression. SVR didn't work well and it was very slow to train. Thus, we didn't include it in this report.

Our best model gives us top 30% in the Kaggle leaderboard, which means there's still a lot of space for improvement. Actually, the most obvious one is using more data. We have more than 55 million rows in the dataset but we only used 1 million of them. We weren't able to use more due to the limitations of time and hardware. Besides, more data preprocessing techniques and feature engineering tricks can be deployed for better performance.

5 Acknowledgments

Our deepest gratitude goes first and foremost to Dr.Gerard de Melo, instructor of massive data storage and retrieval. His profound knowledge and interesting lectures inspired our experiments. He is so kind that he put back the deadline to allow us to fully improve our experiments and reports during the busy exam weeks.

High tribute shall be paid to the authors of the following library:

- **Machine Learning libraries:** sci-kit learn,

Model	Mean Absolute Error	Root Mean Square Error	Kaggle Score
LightGBM	1.5536	3.4178	3.0651
Xgboost	1.6521	3.7539	3.6441
Random Forests	2.1236	4.0966	3.6452
MLP	1.6610	3.8240	3.2783

Table 3: Results for all models

tensorflow, lightgbm

- **Basic Implementation libraries:** numpy, pandas, matplotlib, scipy, seaborn

Thanks to these outstanding pioneers in the field of machine learning, we can easily implement so many machine learning algorithms now.

Last but not least, our thanks go to our beloved families for their loving considerations and support all through these years. We also owe our sincere gratitude to our friends and fellow classmates who gave us their help.

A Feature importance

We present top features of different models in Figure 8, 9 and 10. As we can see, LightGBM utilizes more features while the Random Forests utilizes nearly only distance. This might be the reason that LightGBM has a great performance while the Random Forests performs the worst.

References

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3149–3157, USA. Curran Associates Inc.

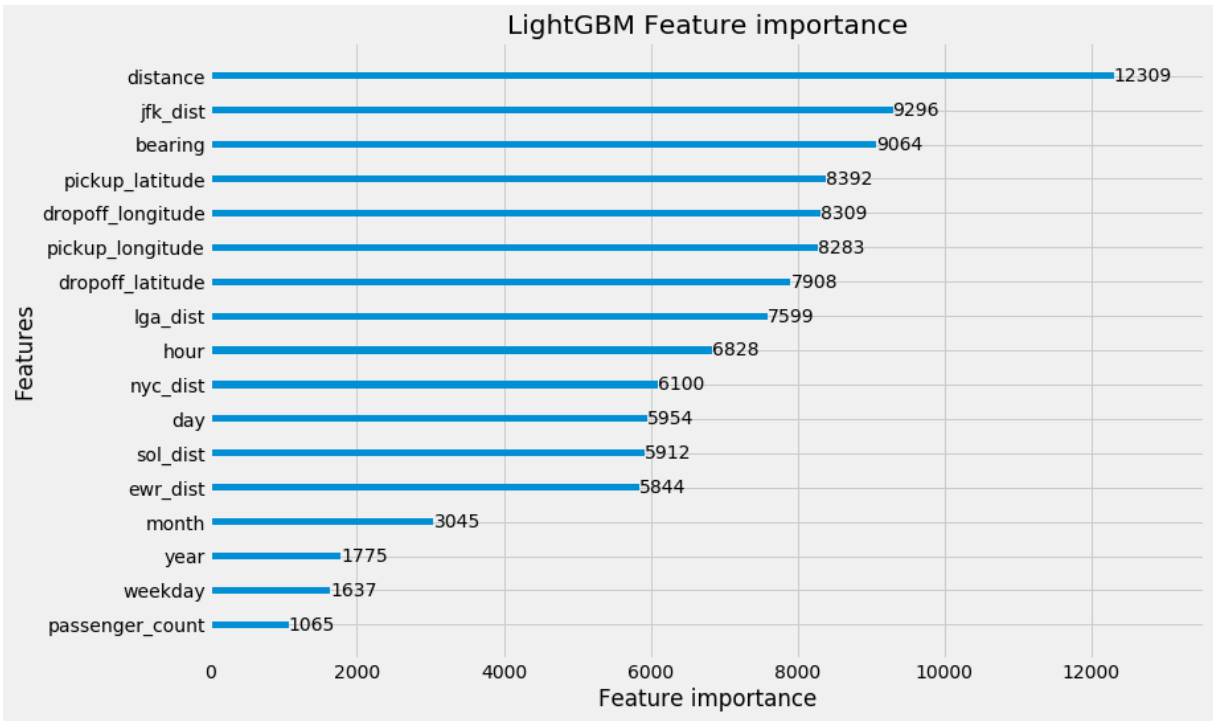

Figure 8: Top features of LightGBM

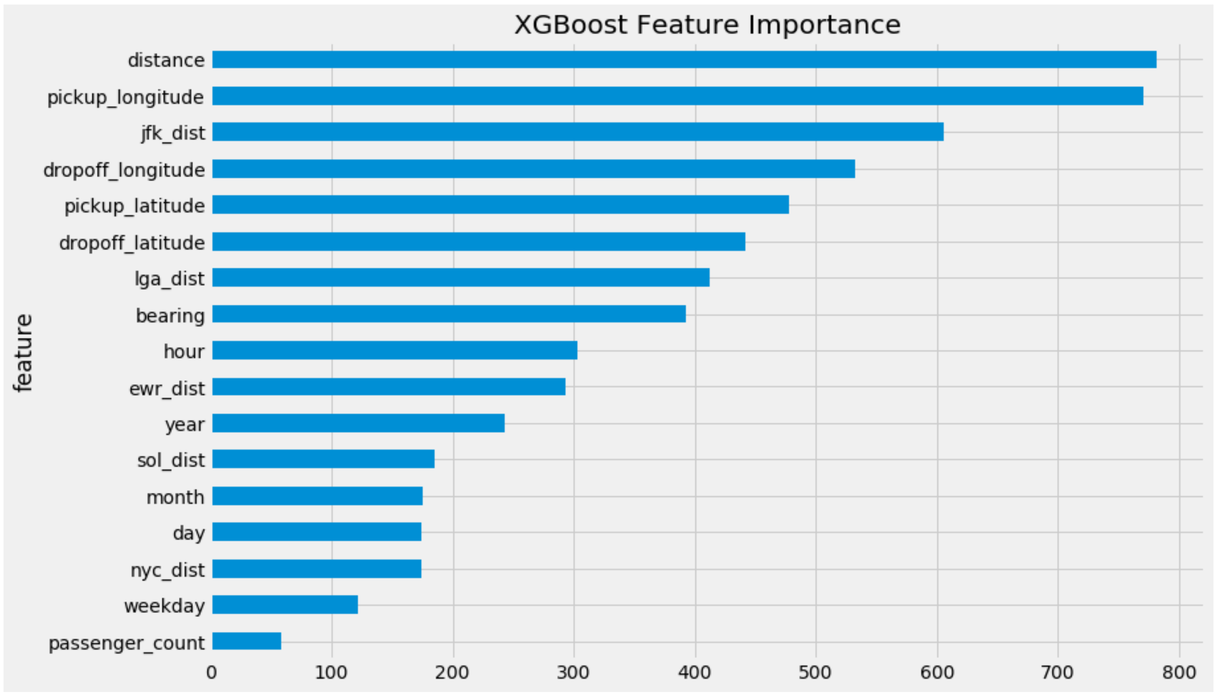

Figure 9: Top features of XGBoost

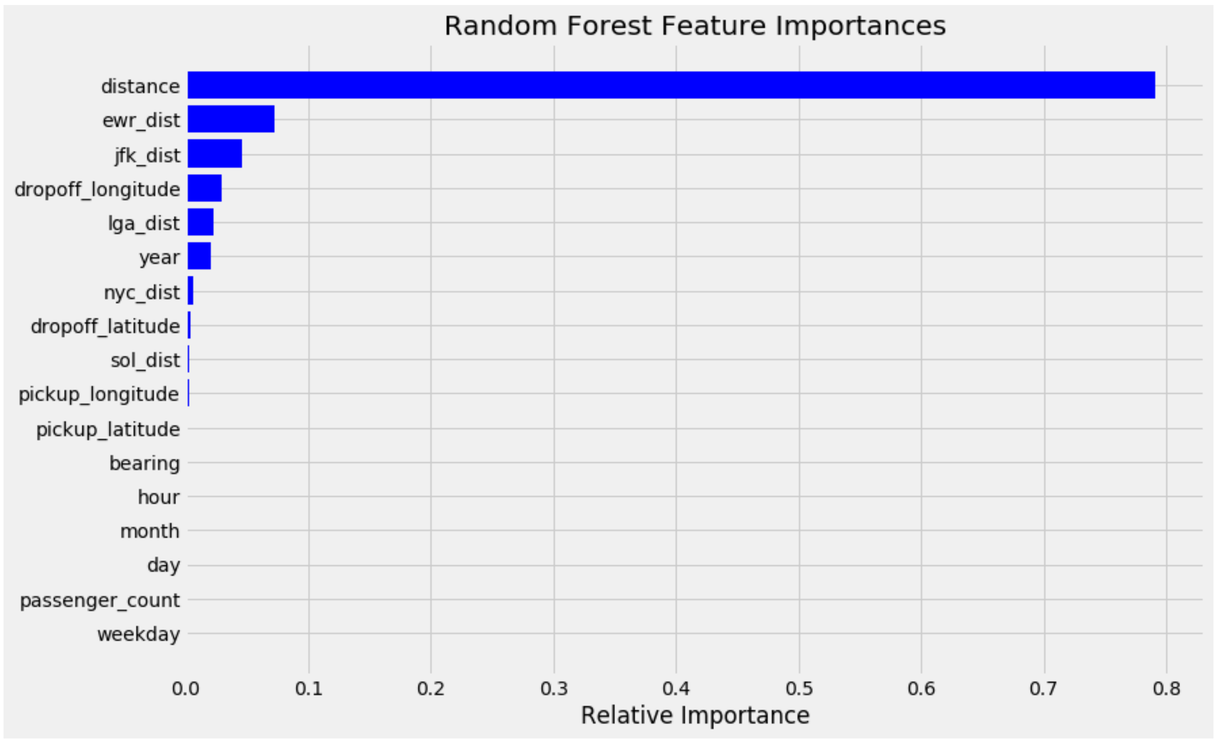

Figure 10: Top features of Random Forests