Using SparkML to Predict New York City Taxi Fare

Group Jet Lag:
Meng-Ting(Joyce) Chang, Jenny Kong, Xinke Sun, Zhe Yuan

About the Data

The data we chose includes basic information about taxi fares in New York City form the year 2009 to 2015. The training set includes six features: pickup_datetime, pickup/dropoff longitude, and latitude. With 55,423,855 rows of record, we have an over 5GB csv file, which is way too big and time-consuming for local computing. Thus it is a good implementation of MongoDB to store and process.

Analytics Goal

Our target is to predict the fare amount of each taxi trip use SparkML packages.
We chose this data because we think this is a typical case where we can learn to work as a group on how to handle large datasets through AWS S3 bucket, store it in MongoDB (through EC2) with ease and apply machine learning models using SparkML under the distributed computing environment. We also launch multiple EMRs to compare our execution times. Moreover, this is an interesting real-world problem that our team is curious to discover and explore more.

Data Processing and Feature Engineering

Our dataset is more than 5GB in total after re-formatting. In order to let our group member have easy access to the dataset, We processed it by uploading it to the AWS S3 and storing in MongoDB through EC2. In EC2 we created ten instances in total, 2 shards with 3 replicates in each shard. Also, we have a configuration which specify how data were stored in which shards. Mongos can talk to dataset through the configurations. Finally, we installed the Jupyter Notebook and apply our analysis through different sizes of clusters on EMR. The two different EMR clusters we have tried are an extra-large with four nods and another extra-large with 3 nods.

Now since we have loaded data and set up the environment successfully, we could focus on how to analyze the dataset we have. By searching New York City taxi fare rules[1] and standards[2], we have a better understanding of its fare structure, so that we can better choose and sort, and manipulate the data according to their importance in fare prediction.

After doing some research on how the taxi fare is calculated, We noticed that some of the data presents might not correct. First, we drop all rows of data with fare-amount less than five dollars, we believe those records are either incomplete or misleading. Then we check on the pick up as well as drop off location longitude and latitude. We want to focus on the fare prediction within

[1] NY City Yellow Taxi Fare Rate: https://www1.nyc.gov/nyc-resources/service/1271/yellow-taxi-fares
[2] NY City Taxi Rate of Fair: http://www.nyc.gov/html/tlc/html/passenger/taxicab_rate.shtml

New York City and we decide to drop all rows of data if they include longitude or latitude out of the NY city range. For example, we drop records of trips that starting from NY city but end in New Jersey area, or Long Island. Or if a trip is long distances from the surrounding areas all the way come into the city, we do not count them into our data. We do this, expecting to improve the reliability of prediction, since if the parts of the trip ends up/starts from out of the city the fare calculation is most likely to be different from the ones that are within the city area.

Using all existing longitude and latitude pairs, we define a function to calculate the great circle distance[3] between the pair specified in decimal degrees. We call this value 'distance' and use it to replace the longitude and latitude.

Moreover, we also notice that date time is an important feature for us, and we can get crucial information from it. We decided to split this column into the year, month, the day of the week, and the time of the day. Furthermore, in order to simplify our calculation while do not sacrifice too much accuracy, we try to reformat month into the quarter of the year and reformat the time of the day to rush hours/no rush hours. We abstract all those information, implement one-hot encoding to avoid unnecessary ordering/sorting. We end up with piping all those columns into a 'features' vector, which is a simple, clean dataset easy to be used for training with different machine learning models.

Machine Learning Algorithm and Evaluation

Since we are performing prediction on the fare amount of taxi ride, we need to implement regression models other than classification models. The algorithms we chose from SparkML library, which is somehow limited in the parallel computing structure, are linear regression, decision tree, and random forest. The evaluation metric we picked is the Residual Mean Squared Error, which in this context could be explained as the mean error(the difference between predicted and actual fare amount).

Linear Regression

First, we try the simple linear regression model. Here X is the 'features' vector, and Y is our predicted fare value.

$$Y = X\beta_1 + \beta_0$$

The residual sum of squares (RMSE) is \$6.02, and R-square is 58.57%, which is fairly good. This result indicates that our prediction is around \$6 deviation with our true value, and almost two-thirds of the data can be explained by our model. But we believe we could do better by applying other non-linear models to capture more complicated interactive-relations within the data.

[3] Refer to:
https://stackoverflow.com/questions/15736995/how-can-i-quickly-estimate-the-distance-between-two-latitude-longitude-points

Decision Tree
We choose decision tree regression because it is fast, easy to integrate, and performs well on the large dataset. The RMSE is $5.89, better than linear regression. We also try to improve the model performance and reduce the possibility of overfitting issues. We try to tune parameters but the results are not obvious enough, while it takes much longer to run the model. End up we just use all the default parameters.

Random Forest
We used the random forest because it is a strong modeling technique and much more robust than a single decision tree. A random forest regression can largely eliminate the possibility of overfitting issue caused by the single decision tree model. Fit our data into the random forest model we get the RMSE $5.27.

Lesson Learned through the Project

One of the most important things we learned through this project is how to deal with large dataset using AWS. By installing multiple shards replicates on EC2 we can compute faster and quicker through the distributed system. Once it launches successfully, the model could be run more efficiently.

The second thing we learn through the project is that we have to clearly know the data type of each column in the data frame. The dataset contains information related to time, which is the string type. In order to implement it into the machine learning models in SparkML, we need to change them to the numeric type of data.

Another thing we think about is the feature engineering procedure. For the feature ‘distance’ our calculation based solely on the differences between the start and end points, but maybe the actual routes, indicating where in New York City the taxi is traveling, would be much more useful and precise. However, this requires us to mapping all location points to a actual city map, and then calculate routes for trips. The workload is large and we would love to try it if we have more times.

Through this fun project, we gained better knowledge about how a distributed system could help us with large dataset computing and how to apply machine learning models in SparkML. We will keep digging and explore more in the future.