Capstone Project

Machine Learning Engineer Nanodegree
New York City Taxi Trip Duration
Wenhan Ji
Dec. 10th, 2017

Definition

Project Overview

Machine learning is a quite popular topic recent days. With more advanced algorithms being created and higher computing ability, we can train the machine learning model based on history data to predict feature with high accuracy. This kind of predictive model can be applied into diverse domains such as bio-informatics, economy, electrical engineering so as to optimize industrial products or help to make decisions.
In this project, I trained a regression model using XGBoost algorithms to predicts the total ride duration of taxi trips in New York City. Once this model gets applied, the NYC taxi drivers can know how long the customers would take the taxi in advance with several features get detected. The primary dataset is one released by the NYC Taxi and Limousine Commission, which includes pickup time, geo-coordinates, number of passengers, and several other variables. In addition, I used NYC 2016 weather dataset to add weather related features to each samples, including temperature, precipitation, snow fall and snow depth.

Problem Statement

The goal is to create a regression model to predict NYC taxi trip duration; the tasks involved are the following:
1. Download 2016 NYC Yellow Cab trip record data and 2016 NYC weather data
2. Do the feature engineering work that create some new features based on existing features.
3. Visualize different features distribution to do the exploratory data analysis and data cleaning.
4. Define loss function and split training data and testing data.
5. Train XGBoost model and visualize the feature importance.

Metrics

Accuracy is Root Mean Squared Logarithmic Error.

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:
ϵ is the RMSLE value (score)
n is the total number of observations in the (public/private) data set,
pi is your prediction of trip duration, and
ai is the actual trip duration for i.
log(x) is the natural logarithm of x

Analysis

Data Exploration

1. The nyc_taxi_trip_duration dataset has 11 features and 1458643 samples.

```
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 11 columns):
id                    1458644 non-null object
vendor_id             1458644 non-null int64
pickup_datetime       1458644 non-null datetime64[ns]
dropoff_datetime      1458644 non-null datetime64[ns]
passenger_count       1458644 non-null int64
pickup_longitude      1458644 non-null float64
pickup_latitude       1458644 non-null float64
dropoff_longitude     1458644 non-null float64
dropoff_latitude      1458644 non-null float64
store_and_fwd_flag    1458644 non-null object
trip_duration         1458644 non-null int64
dtypes: datetime64[ns](2), float64(4), int64(3), object(2)
memory usage: 122.4+ MB
```

- id - a unique identifier for each trip
- vendor_id - a code indicating the provider associated with the trip record
- pickup_datetime - date and time when the meter was engaged
- dropoff_datetime - date and time when the meter was disengaged
- passenger_count - the number of passengers in the vehicle (driver entered value)
- pickup_longitude - the longitude where the meter was engaged
- pickup_latitude - the latitude where the meter was engaged
- dropoff_longitude - the longitude where the meter was disengaged
- dropoff_latitude - the latitude where the meter was disengaged
- store_and_fwd_flag - This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
- trip_duration - duration of the trip in seconds

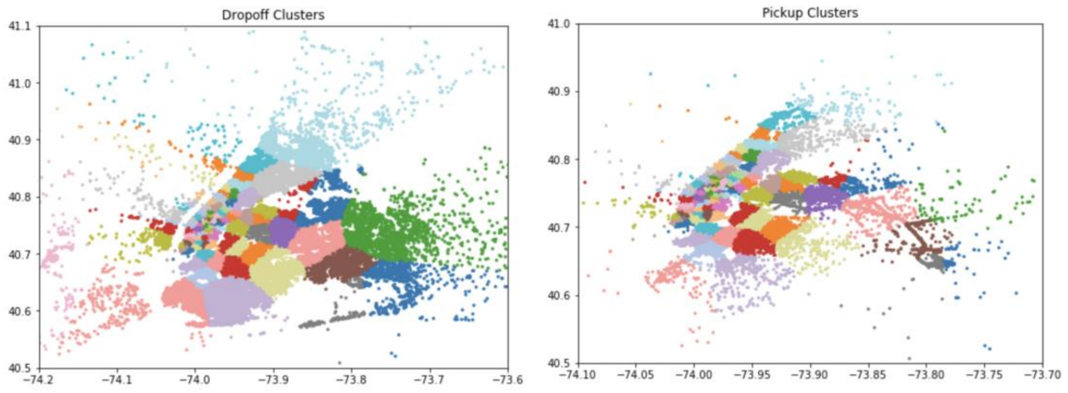
2. The 2016nyc_weather dataset has 7 features and 366 samples.

```
RangeIndex: 366 entries, 0 to 365
Data columns (total 7 columns):
date                     366 non-null datetime64[ns]
maximum temerature       366 non-null int64
minimum temperature      366 non-null int64
average temperature      366 non-null float64
precipitation            366 non-null object
snow fall                366 non-null object
snow depth               366 non-null object
dtypes: datetime64[ns](1), float64(1), int64(2), object(3)
memory usage: 20.1+ KB
```

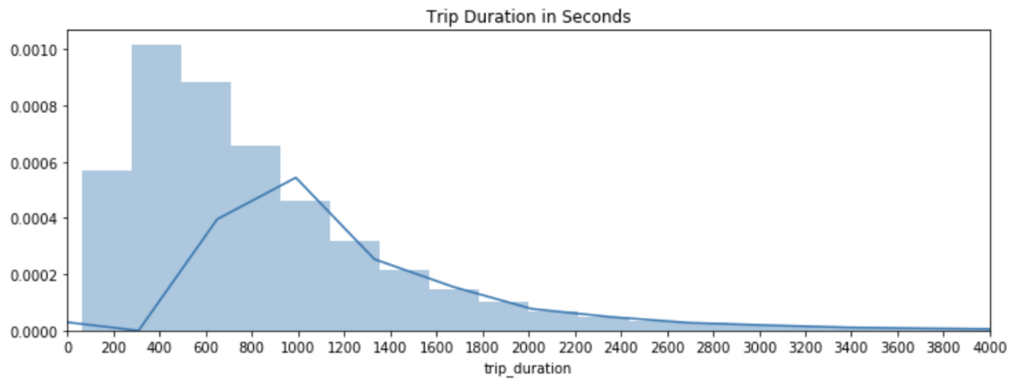
Weather data collected from the National Weather Service. It contains the first six months of 2016, for a weather station in central park. It contains for each day the minimum temperature, maximum temperature, average temperature, precipitation, new snow fall, and current snow depth. The temperature is measured in Fahrenheit and the depth is measured in inches. T means that there is a trace of precipitation.

Data Visualization (After Feature Engineering and Data Cleaning)

1. The below two plot visualize dropoff_clusters and pickup_clusters using scatter plot at corresponding location. These two features are computed by mini batch kmeans algorithm with n_clusters set to be 100. From the plots, we can find that this clustering makes sense with data distributes uniformly in each cluster.

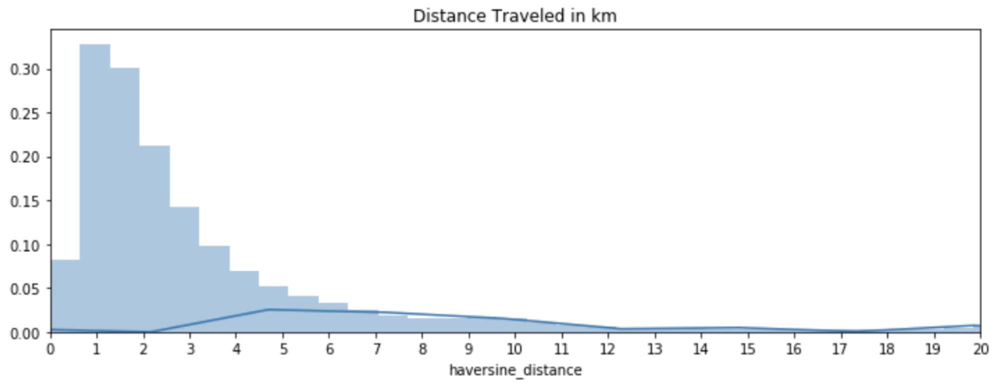


2. From the below histogram of trip duration, we will know that most trip duration is around 11 minutes.



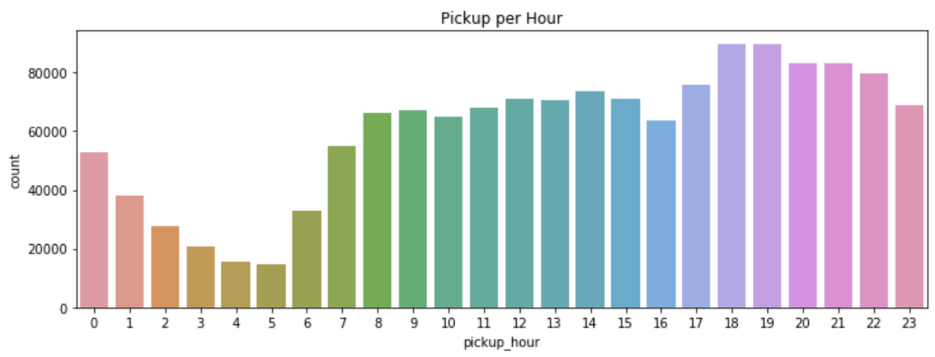
```
Mean trip duration : 14.07 min
Max trip duration : 719.00 min
Min trip duration : 1.00 min
Median trip duration : 11.10 min
```

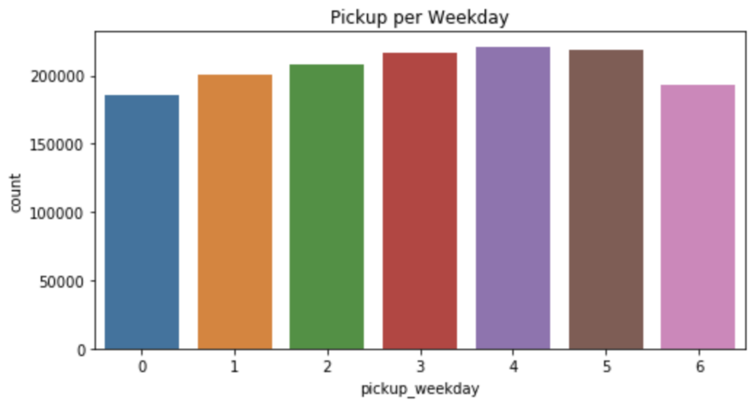
3. From the below histogram of haversine_distance, we will know that most trip's haversine distance is around 2km.

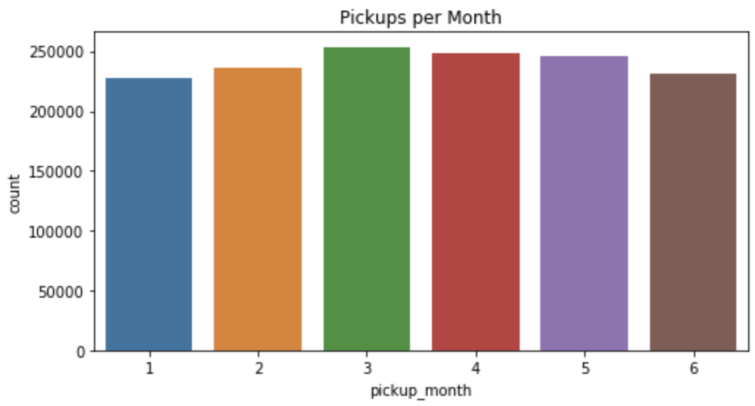


```
Mean trip distance (km): 3.47
Max trip distance (km) 320.13
Median trip distance (km): 2.11
```

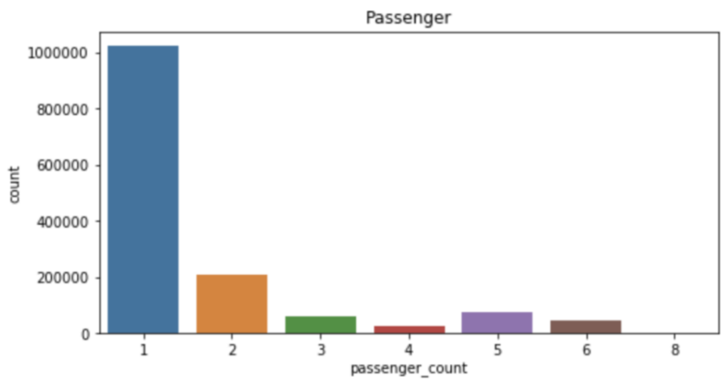
4. From the below count plot of pickup hour, pickup weekday, pickup month, we will know that at around the hour of 18, people's need to take a taxi reached the peak. Maybe it's the time to get off work. The count of people take taxi at weekday of 4 is a little bit higher than other days. The count of people take taxi at month of 3 is a little bit higher than other weeks.



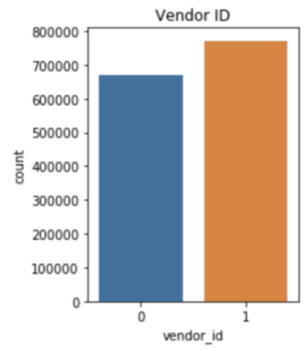




5. From the below count plot of vendor id, we can see that most of time, there's only one passenger that takes the taxi.



6. From the below count plot of vendor id, we can see the trip provider 1's records count is more than provider 2's.

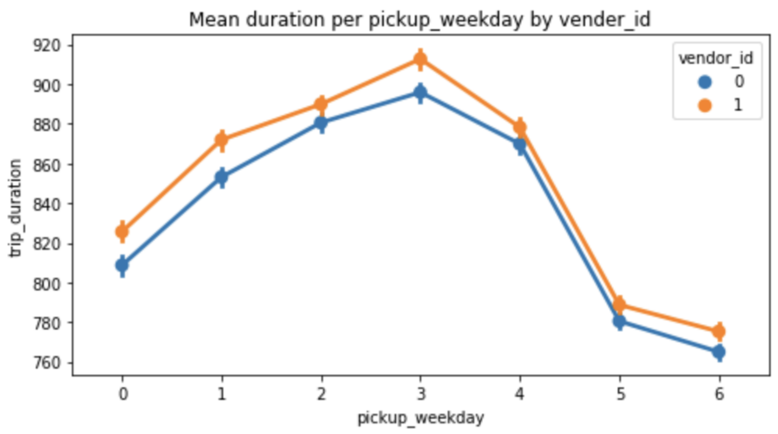


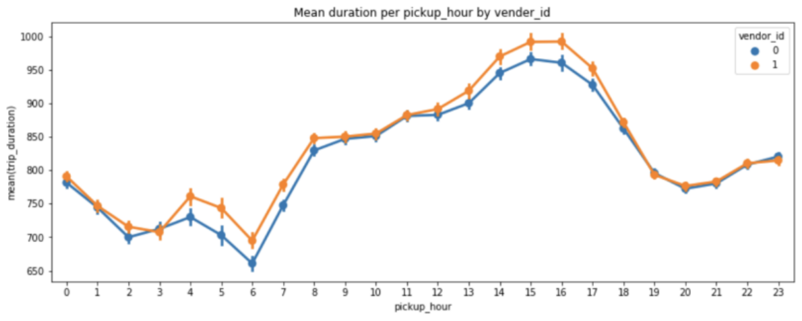
6. From the below point plot of mean trip duration, we can get several observations.

6.1 Trip records from vendor 1 has a higher mean trip duration then vendor 2's records.

6.2 At weekday of 3, people's taxi mean trip duration is the highest. At weekday of 6, people's taxi mean trip duration is the lowest.

6.3 At around the hour of 16, people's taxi mean trip duration is the highest. At hour of 6, people's taxi mean trip duration is the lowest.





Algorithms and Techniques

1. The machine learning algorithm that cluster dropoff longitude, dropoff latitude, pickup longitude, pickup latitude features is Mini Batch KMeans. In this way, I created pickup_cluster

and dropoff_cluster two new features. This algorithm can be seen below.

```
Algorithm 1 Mini Batch K-Means algorithm
  Given: k, mini-batch size b, iterations t, data set X
  Initialize each c ∈ C with an x picked randomly from X
  v ← 0
  for i ← 1 to t do
      M ← b examples picked randomly from X
      for x ∈ M do
          d[x] ← f(C,x)
      end
      for x ∈ M do
          c ← d[x]
          v[c] ← v[c] + 1
          $\eta \leftarrow \frac{1}{v[c]}$
          c ← (1-$\eta$)c+$\eta$x
      end
  end
```

2. Also, with dropoff longitude, dropoff latitude, pickup longitude, pickup latitude features, I computed haversine distance. The formula can be seen below.

$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

where

- hav is the haversine function:

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1-\cos(\theta)}{2}$$

- d is the distance between the two points (along a great circle of the sphere; see spherical distance),
- r is the radius of the sphere,
- φ_1, φ_2: latitude of point 1 and latitude of point 2, in radians
- λ_1, λ_2: longitude of point 1 and longitude of point 2, in radians

3.The predictive regression algorithm that train the feature data so as to predict trip duration is XGBoost.

XGBoost is an open-source software library which provides the gradient boosting framework for C++, Java, Python, R, and Julia. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library".

XGBoost is short for “Extreme Gradient Boosting”, where the term “Gradient Boosting” is proposed in the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. XGBoost is based on this original model. This is a tutorial on gradient boosted trees, and most of the content is based on these slides by the author of xgboost.

Benchmark

To create an initial benchmark for the classifier, I used scikit-learn library, to try multiple regression algorithms. The XGBoost algorithm achieved the best accuracy, around 0.8.

Methodology

Data Preprocessing

The preprocessing done in the 'Feature Engineering' and 'Data Cleaning' part of notebook consists of the following steps:

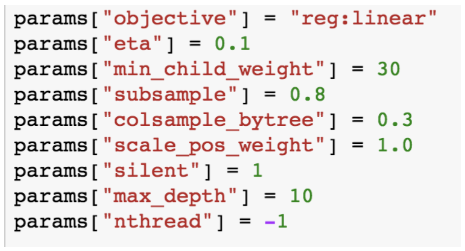
1. Create pickup_cluster feature using Mini Batch KMeans algorithms with the help of 4 location

features: pickup_latitude, pickup_longitude, dropoff_latitude, dropoff_longitude.
2. Compute haversine_distance feature using haversine package with pickup and dropoff location information.
3. Parse out date, weekday, hour, month, day feature from pickup_datetime feature.
4. Merge NYC taxi duration kaggle dataset and 2016 NYC whether dataset based on the date feature.
5. Encode the categorical feature store_and_fwd_flag and vendor_id to 0,1.
6. Clean datasets based on the visualization plots and computed max, min, mean parameters of each feature. I deleted records that trip duration is less than 1 minutes and over 12 hours, trip haversine distance less than 0.01km and over 400km. These deleted records do not make sense, as the trip is too short or too long.
7. Split the training and testing data and test set size is 20% of original dataset.

Implementation

Benchmark model from sklearn:
1. linear model
2. K Neighbors Regressor with n_neighbors=5
3. Gradient Boosting Regressor with n_estimators=100

XGBoost model from official XGBoost document:
Parameters:



```
params["objective"] = "reg:linear"
params["eta"] = 0.1
params["min_child_weight"] = 30
params["subsample"] = 0.8
params["colsample_bytree"] = 0.3
params["scale_pos_weight"] = 1.0
params["silent"] = 1
params["max_depth"] = 10
params["nthread"] = -1
```

Results

Model Evaluation

Benchmark model from sklearn:
1. linear model: Root Mean Squared Logarithmic Error= 0.074
2. K Neighbors Regressor: Root Mean Squared Logarithmic Error= 0.073
3. Gradient Boosting Regressor: Root Mean Squared Logarithmic Error= 0.052

XGBoost model:
Root Mean Squared Logarithmic Error= 0.044

Conclusion

Reflection

The process used for this project can be summarized using the following steps:
1. An initial problem and relevant, public datasets were found
2. The data was downloaded
3. Compute and create new features based on existing features
4. Visualize features to do the exploratory data analysis and data cleaning
5. Train test split and the test data function as the future unknown data.
6. Benchmark model was created for the regressor
7. The regressor was trained using the data

8. Do the model evaluation based on test set and loss function.

I found steps 3 the most difficult. The first time I faced the dropoff and pickup location data, I have no idea how these features can help me improve my model. It's after I read some documents then know they can help to compute the haversine distance. And the KMeans clustering can be applied to cluster the location.

The visualization part is also import that the plots help me have a deeper insight of different features distribution so that more information can be extracted.

Compared with benchmark model, XGBoost model show a better performance to do this regression task.