Taxi Trip Time Prediction Report

Wei Guo, guowei123@gmail.com

# Feature engineering

Features used in models

* Continuous
  + Haversine distance
  + Avg\_Temperature
  + Prescription
  + New\_Snow
  + Snow\_Depth
  + DistanceToDowntown
  + DirectionToDowntown
  + trip\_cnt\_by\_start
  + trip\_cnt\_by\_end
  + trip\_duration\_by\_start
  + trip\_duration\_by\_end
* Categorical
  + start\_month
  + start\_dayOfWeek
  + start\_hour
  + start\_isWeekend
  + start\_isHoliday
  + geoDist

Observations

* **Haversine distance** is the most predictive feature and dominates all other features.
* I created feature **geoDist**, which is based on the **geohash** of starting and ending coordinates, I wanted to see if it is better than Haversine distance, unfortunately it is not.
* I rounded up longtitude and latitude to three decimal points and count the number of trips as well as the average duration of the trips in each unique long/latitude combination. Two potential benefits:
  + I can find out the “**hot spot**” by the number of trips in each long/lat for both starting and ending points.
  + Since long/lat are **high cardinality categorical features**, doing this and creating a lookup table is a way to convert high-cardinality feature to continuous ones, otherwise they can’t be used directly with so many levels.
* I created features **DistanceToDowntown**, which is the Haversine distance from starting point to the most visited destinations (we treat it as center of the city or downtown).
* I created features **DirectionToDowntown**, which is the direction(angle) from starting point to downtown. It indicates the trip is heading to downtown or heading from downtown.
* I downloaded daily **weather data** from [weather.gov](http://w2.weather.gov/climate/xmacis.php?wfo=okx), hoping that using weather info can improve model’s performance, but it doesn’t much. Possible reasons might be that the weather is on daily basis, if we have hourly weather info in real time, it might be more helpful.
* To catch the **rush hour and rush day**, I created several features, i.e., start\_month, start\_dayOfWeek, start\_hour, start\_isWeekend, start\_isHoliday, they reasonably well as we expect.
* Some **extreme trip durations**, especially those less than 60 seconds brings some in-stability of the model.

# Modeling

I have tried the following models and found that GBM is the best:

* Linear Model with Continuous features
* Linear Model with ScaledContinuous features
* Linear Model with categorical features
* Linear Model with ScaledContinuous and categorical features
* Linear Model with ScaledContinuous and categorical features with regularization
* Random Forest
* GBM

|  |  |
| --- | --- |
| Model | RMSE in validation (in seconds) |
| Linear Model with Continuous features | 568.68 |
| Linear Model with ScaledContinuous features | 568.89 |
| Linear Model with categorical features | 666.77 |
| Linear Model with ScaledContinuous and categorical features | 550.94 |
| Linear Model with ScaledContinuous and categorical features with regularization |  |
| Random Forest (max\_depth=5, random\_state=0, n\_estimators=75) | 554.66 |
| GBM {'n\_estimators': 100, 'max\_depth': 3, 'min\_samples\_split': 5, 'learning\_rate': 0.1, 'loss': 'ls'} | 538.73 |

# Potential improvement

* Features to try
  + More accurate weather info: hourly update
  + Real time speed of the vehicle
  + Driver info: history of driving (accidents, tickets, claims, ratings)
  + Rider info: history usage (average trip duration/distance, local resident or not)
  + Vehicle info: make, age
  + Route traffic status: congestion, accidents, construction blocks
  + Whether the trip is Highway or local
  + Speed limit on the route, especially lower ones
* Modeling
  + Fine tune model hyper-parameters
  + Ensemble several models
  + Try XGBoost (ran out of memory in my machine)
  + Try Deep Neural Nets
  + Fine tune features: transformation, proper binning
  + Tried cross-validation but out of memory, did train-test split instead

# Real time implements

* The scoring algorithm must be very fast, algorithms like KNN should be avoided since it takes long time to lookup in the training data to find similar points.
* For features that need to look up values in some “look-up” table, the look-up table has to be cached in advance and accessible quickly.
* If API call is needed to create features, the API’s latency has to be considered. The ability of processing huge volume of calls needs to be verified.
* Since new data is accumulated very fast, how often and in what method should the model be updated. There are several choices such as batch learning, SGD to consider.

# Others