PREDICTION OF DESTINATION OF VEHICLES

Ziqing Zhou, Chen Qiu

zhou.ziq@husky.neu.edu qiu.che@husky.neu.edu

INFO 7390, Fall 2018, Northeastern University

1. Abstract

A study by a research team at Northeastern University in the United States shows that 93% of human behavior is predictable. If you think about our regular life, you can feel that this number is not so exaggerated. It is precisely because of this that traffic can be rationally planned and the city can be developed in an orderly manner. In practical applications, it is not so easy to predict, and the lack of data is an important reason. So to what extent can we predict people's behavior under limited data?

In this paper, we developed a model to predict the destination of a vehicle based on history route. Our project is from a competition, powered by DiDi(a Chinese ride-sharing, artificial intelligence and autonomous technology conglomerate) and Data Casatle(data competition platforms) , We enrolled in this competition and there's another 1017 teams compete with us. model is trained to predict the destination of the vehicle during the trip given the vehicle id, time and departure location. There are two ways to do the prediction, one is using regression and the other is using classification. we deployed several methodologies like DBSCAN cluster, Gradient Boosting(XGB) and tested the dataset on each of them. After comparing each of these models and their accuracies with the teams that already submit their result, we found that our model would rank top 20 among all the teams.

This report contains an introduction to the problem and comprehensive analysis on the dataset followed by feature engineering. Then we applied our classification models got the score with 0.39

2. Introduction

Recently, auto mobile is a hot topic in high-tech area, Tesla, Uber and Waymo are busing with improving the technology on auto driving. Well, what about the car knows where you would go the moment you sit in. the car, based on the user's history route, time when you leave and arrive as well as the coordinate, cars will learn and train the model to predict the destination, in this way auto mobile will be brought into a higher level to improve convenience of people's life.

The dataset provided by the host of competition including a test.csv dataset and a train.csv dataset, unlike some team using logistic prediction by calculate where the vehicles go most often in the days 1-7 days of the week, mark these locations using 1 or 0, and use this as a label to recommend the location with the highest probability to the player and use logistic regression to predict, we add some features and turn the problem into a classification problem. we decide to use the distance between the predict position and the actual destination to evaluate our model, we also translate the data in the dataset into data time in pandas and added two new feature, ‘the week of leave’ and ‘the hour of leave’, also we combined latitude and longitude by turn latitude and longitude to geohash code.

As the dataset contained number of inconsistent and null values which could have hampered the prediction, we performed data cleaning and dropped the irrelevant entries. Further Exploratory Data Analysis (EDA) was performed, resulting into statistical insights and visualizations on the trained dataset

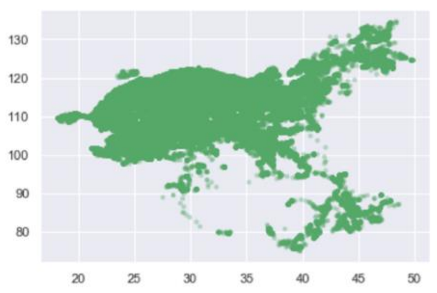
3. Method

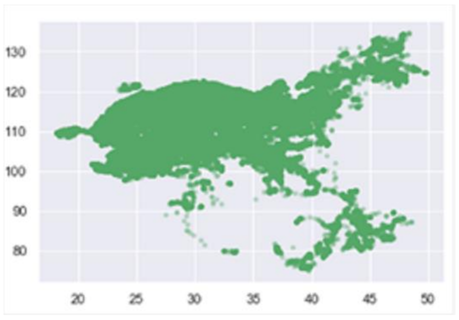
Since our entire dataset is basically points and timestamp, which is the destination and the start point of vehicle, each data entry is basically a pair of an input vector and a desired output value.

3.1 Exploratory Data Analysis

Exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

First, we considered it as a regressor problem because the data is numerical, but if we draw the start point and the destination points.

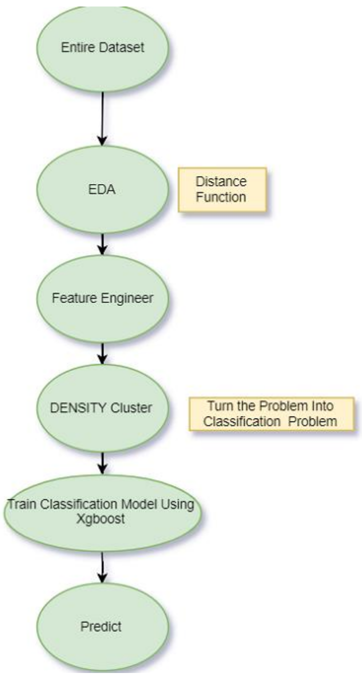




Strat points Destination points

We can hardly see the difference of these two diagrams, and the object of our model is to predict each vehicle’s destination, so we changed our strategy.

Before training the data, need to do some preprocessing work, like define out function to calculate distance using latitude and longitude in the dataset and do some feature engineering work.


Entire Dataset
EDA
Distance Function
Feature Engineer
DENSITY Cluster
Turn the Problem Into Classification Problem
Train Classification Model Using Xgboost
Predict

3.2 Evaluation Function

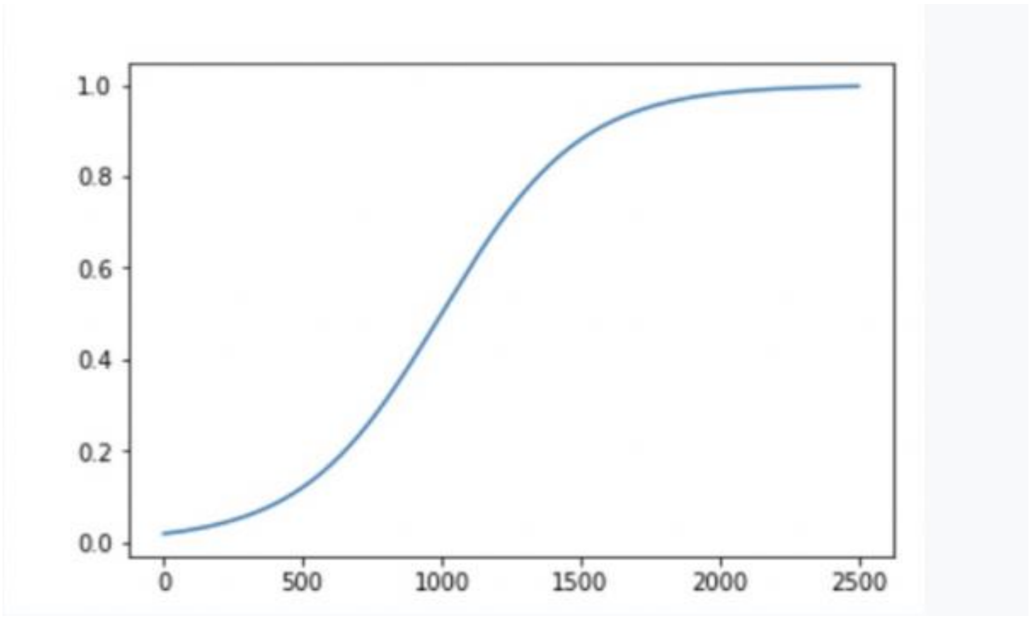
Examine the spherical distance between the predicted destination (latitude and longitude) and the actual destination (latitude and longitude)

Where is the spherical distance (unit: meter) of the predicted destination of the i-th sample and the actual destination, and n is the number of samples in the test set.

$$f(d_i) = \frac{1}{1+\exp\left(-\frac{d_i-1000}{250}\right)} \qquad \text{score} = \frac{1}{n}\sum_{i=1}^{n} f(d_i)$$

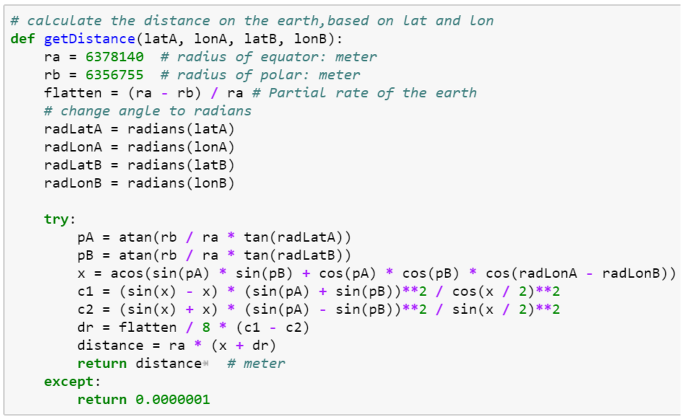
Note: The limit optimal value of score is about 0.01798 (that is, when the prediction error of all destinations is 0 meters)

The relationship between the distance and the score is as shown in the figure below. The horizontal axis is the distance and the vertical axis is the score.



3.3 Distance Function

First of all, we get the longitude and latitude， but we can’t just use Euclidean Distance to calculate the distance of two point, it may cause more error and noise in our calculation, and according to Geographic coordinate system [1], we defined our own distance function to calculate the distance of two points.



```python
# calculate the distance on the earth,based on lat and lon
def getDistance(latA, lonA, latB, lonB):
    ra = 6378140  # radius of equator: meter
    rb = 6356755  # radius of polar: meter
    flatten = (ra - rb) / ra # Partial rate of the earth
    # change angle to radians
    radLatA = radians(latA)
    radLonA = radians(lonA)
    radLatB = radians(latB)
    radLonB = radians(lonB)

    try:
        pA = atan(rb / ra * tan(radLatA))
        pB = atan(rb / ra * tan(radLatB))
        x = acos(sin(pA) * sin(pB) + cos(pA) * cos(pB) * cos(radLonA - radLonB))
        c1 = (sin(x) - x) * (sin(pA) + sin(pB))**2 / cos(x / 2)**2
        c2 = (sin(x) + x) * (sin(pA) - sin(pB))**2 / sin(x / 2)**2
        dr = flatten / 8 * (c1 - c2)
        distance = ra * (x + dr)
        return distance  # meter
    except:
        return 0.0000001
```

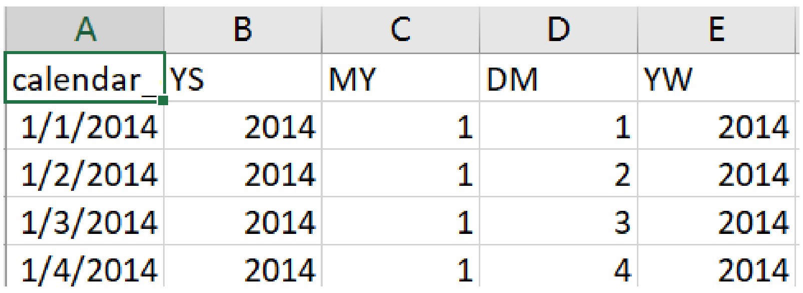
3.4 Feature engineering

The column of the provided data is just about two points and a timestamp,so we need to extend the feature to train out model.

- **Time Feature**

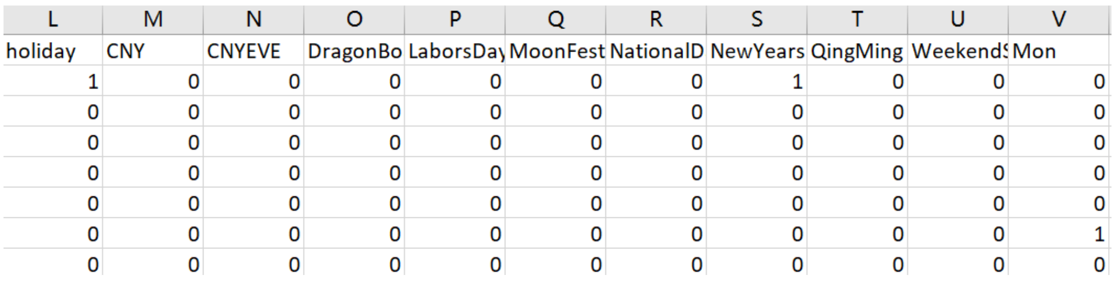
  We made a “Time_feature” table to record the feature of the time,

  First is the numerical information of the data stamp, we saved it as



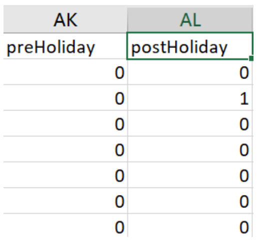
A	B	C	D	E
calendar_	YS	MY	DM	YW
1/1/2014	2014	1	1	2014
1/2/2014	2014	1	2	2014
1/3/2014	2014	1	3	2014
1/4/2014	2014	1	4	2014

And then we use one-hot encoding to mark the holiday of each day,



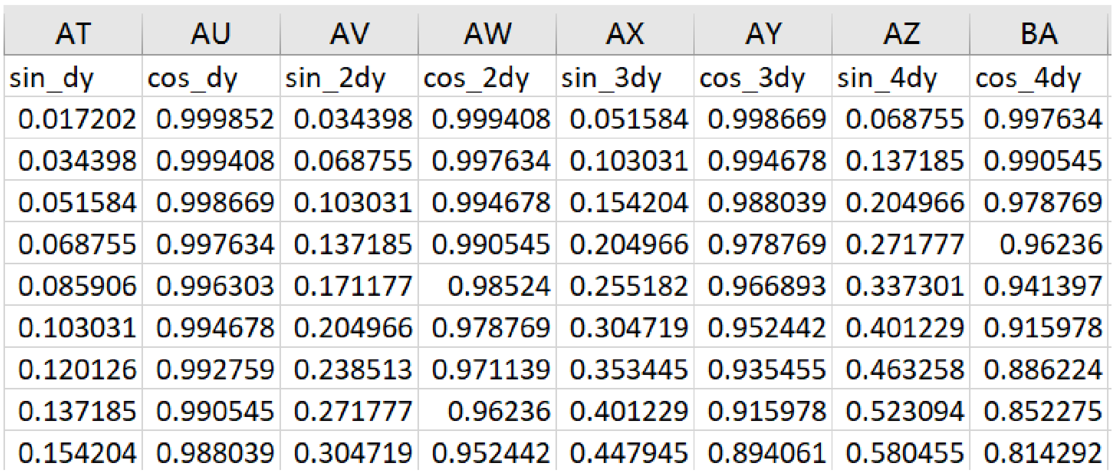
L	M	N	O	P	Q	R	S	T	U	V
holiday	CNY	CNYEVE	DragonBo	LaborsDay	MoonFest	NationalD	NewYears	QingMing	Weekend:	Mon
1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0

And then we consider that the holiday may affect the people out frequent, so we calculate the day that pre or post from the holiday



AK	AL
preHoliday	postHoliday
0	0
0	1
0	0
0	0
0	0
0	0
0	0

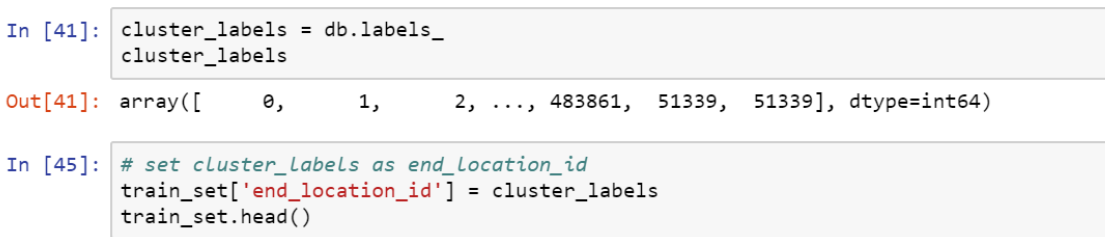
And finally, we calculate the similarity of that day between the holiday, as for here, we use sin/cos to represent it.



AT	AU	AV	AW	AX	AY	AZ	BA
sin_dy	cos_dy	sin_2dy	cos_2dy	sin_3dy	cos_3dy	sin_4dy	cos_4dy
0.017202	0.999852	0.034398	0.999408	0.051584	0.998669	0.068755	0.997634
0.034398	0.999408	0.068755	0.997634	0.103031	0.994678	0.137185	0.990545
0.051584	0.998669	0.103031	0.994678	0.154204	0.988039	0.204966	0.978769
0.068755	0.997634	0.137185	0.990545	0.204966	0.978769	0.271777	0.96236
0.085906	0.996303	0.171177	0.98524	0.255182	0.966893	0.337301	0.941397
0.103031	0.994678	0.204966	0.978769	0.304719	0.952442	0.401229	0.915978
0.120126	0.992759	0.238513	0.971139	0.353445	0.935455	0.463258	0.886224
0.137185	0.990545	0.271777	0.96236	0.401229	0.915978	0.523094	0.852275
0.154204	0.988039	0.304719	0.952442	0.447945	0.894061	0.580455	0.814292

- **Cluster the points as a label**

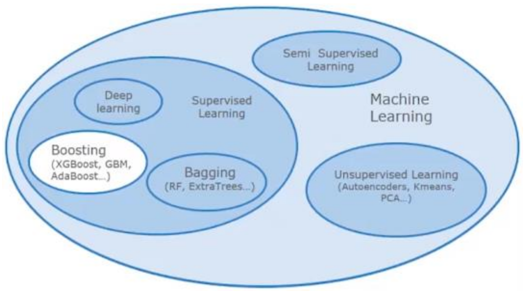
  We used DBscan to cluster the coordinate of destination, and finally, we got 21088 clusters,we added them as a label of each records



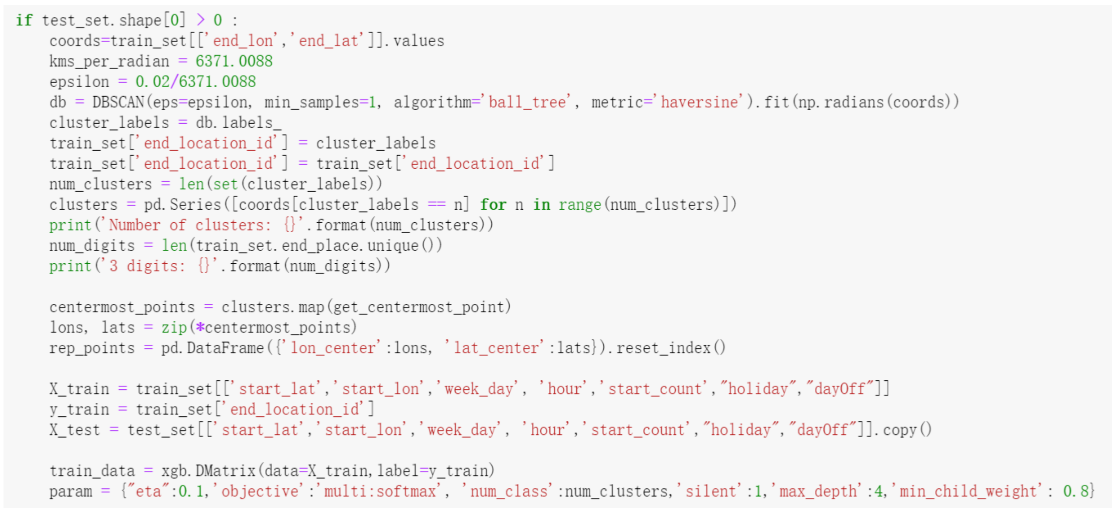
```
In [41]: cluster_labels = db.labels_
         cluster_labels

Out[41]: array([     0,      1,      2, ..., 483861,  51339,  51339], dtype=int64)

In [45]: # set cluster_labels as end_location_id
         train_set['end_location_id'] = cluster_labels
         train_set.head()
```

3.4 XGBoost:

XGBoost (Extreme Gradient Boosting) is developed by ‘Tianqi Chen’.it is an updated gradient tree that to provide a more generalizable Machine Learning model. It’s always performs better than normal GBDT, because it’s done some optimization on multithread process and using the second derivative to evaluate the function. XGBoost optimizes in function space using Newtonian methods. In our project, we use XGBoost to train a classification for each vehicle, and then in the test case, we assume the value into each vehicle’s model. And then make out prediction. Thus, XGBoost could be applied for further validation or mixed with weather data for exploring correlation.


Semi Supervised Learning
Machine Learning
Deep learning
Supervised Learning
Boosting (XGBoost, GBM, AdaBoost...)
Bagging (RF, ExtraTrees...)
Unsupervised Learning (Autoencoders, Kmeans, PCA...)

3.5 DBSCAN



```
if test_set.shape[0] > 0 :
    coords=train_set[['end_lon','end_lat']].values
    kms_per_radian = 6371.0088
    epsilon = 0.02/6371.0088
    db = DBSCAN(eps=epsilon, min_samples=1, algorithm='ball_tree', metric='haversine').fit(np.radians(coords))
    cluster_labels = db.labels_
    train_set['end_location_id'] = cluster_labels
    train_set['end_location_id'] = train_set['end_location_id']
    num_clusters = len(set(cluster_labels))
    clusters = pd.Series([coords[cluster_labels == n] for n in range(num_clusters)])
    print('Number of clusters: {}'.format(num_clusters))
    num_digits = len(train_set.end_place.unique())
    print('3 digits: {}'.format(num_digits))

    centermost_points = clusters.map(get_centermost_point)
    lons, lats = zip(*centermost_points)
    rep_points = pd.DataFrame({'lon_center':lons, 'lat_center':lats}).reset_index()

    X_train = train_set[['start_lat','start_lon','week_day', 'hour','start_count',"holiday","dayOff"]]
    y_train = train_set['end_location_id']
    X_test = test_set[['start_lat','start_lon','week_day', 'hour','start_count',"holiday","dayOff"]].copy()

    train_data = xgb.DMatrix(data=X_train, label=y_train)
    param = {"eta":0.1,'objective':'multi:softmax', 'num_class':num_clusters,'silent':1,'max_depth':4,'min_child_weight': 0.8}
```

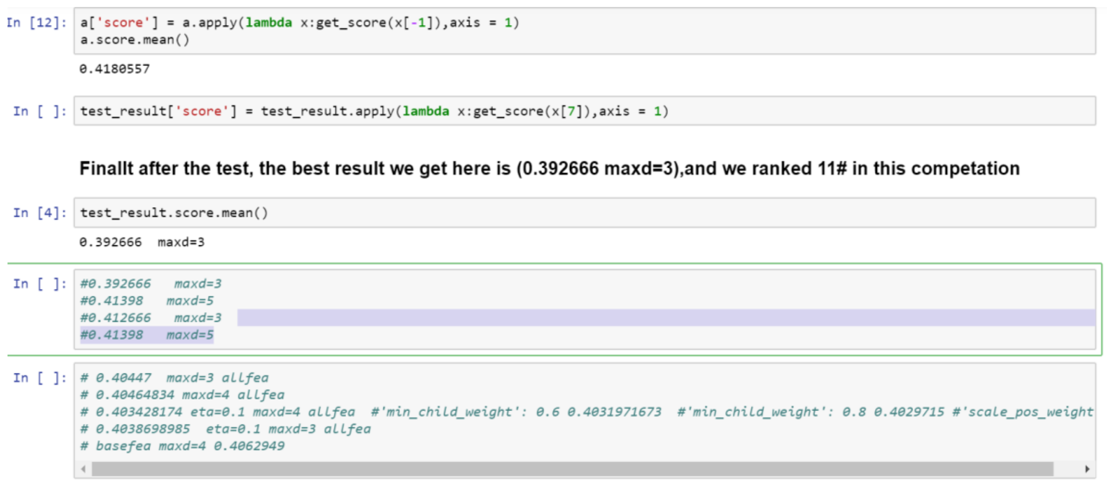
Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996.[1] It is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and most cited in scientific literature.

3.6 Google Colab

Because the calculation power needed here is huge, we need to train 2881 model totally because there's totally 2881 vehicle, and plus windows did not suitable with XGBoost well, So we run our code on the google colab.you can reach read.md for the instruction

4. Result

In our project, by using DBSCAN cluster for the feature



```
In [12]: a['score'] = a.apply(lambda x:get_score(x[-1]),axis = 1)
         a.score.mean()

         0.4180557

In [ ]:  test_result['score'] = test_result.apply(lambda x:get_score(x[7]),axis = 1)
```

Finallt after the test, the best result we get here is (0.392666 maxd=3),and we ranked 11# in this competation

```
In [4]:  test_result.score.mean()

         0.392666   maxd=3

In [ ]:  #0.392666   maxd=3
         #0.41398   maxd=5
         #0.412666   maxd=3
         #0.41398   maxd=5

In [ ]:  # 0.40447  maxd=3 allfea
         # 0.40464834 maxd=4 allfea
         # 0.403428174 eta=0.1 maxd=4 allfea  #'min_child_weight': 0.6 0.4031971673  #'min_child_weight': 0.8 0.4029715 #'scale_pos_weight
         # 0.4038698985  eta=0.1 maxd=3 allfea
         # basefea maxd=4 0.4062949
```

engineering purpose successes fully turn the problem into a classification problem, and we trained a model for each vehicle, can predict the destination of vehicle based on its which cluster it's in and which cluster it may go to, we got a relatively high score in the competition, in a near future, we want to dig deep into the project by using regression or KNN to check whether we can do better.

5. Citations

[1] https://github.com/carlosbkm/car-destination-prediction

[2] http://www.cs.cmu.edu/~coral/old/publinks/brettb/06itsc-driver-intent.pdf

[3] https://www.nbcnews.com/mach/science/elon-musk-says-tesla-s-ai-will-let-cars-predict- ncna813211

[4] https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/

[5] https://www.fastcompany.com/3035350/uber-can-now-predict-where-youre-going-before-you-get-in-the-car

[6] https://pdfs.semanticscholar.org/fd0f/98aa362d6b86f02a28c571e5c40d8d8ff65d.pdf

[7] http://people.cs.aau.dk/~csj/Papers/Files/2006_brilingaiteITSS.pdf

[8] https://www.hindawi.com/journals/mpe/2015/824532/

[9] https://en.wikipedia.org/wiki/History_of_the_automobile

[10] http://deeplearning.net/tutorial/mlp.html

[11]https://warwick.ac.uk/fac/cross_fac/complexity/study/emmcs/outcomes/studentprojects/gandhi.pdf

[12] https://en.wikipedia.org/wiki/Exploratory_data_analysis