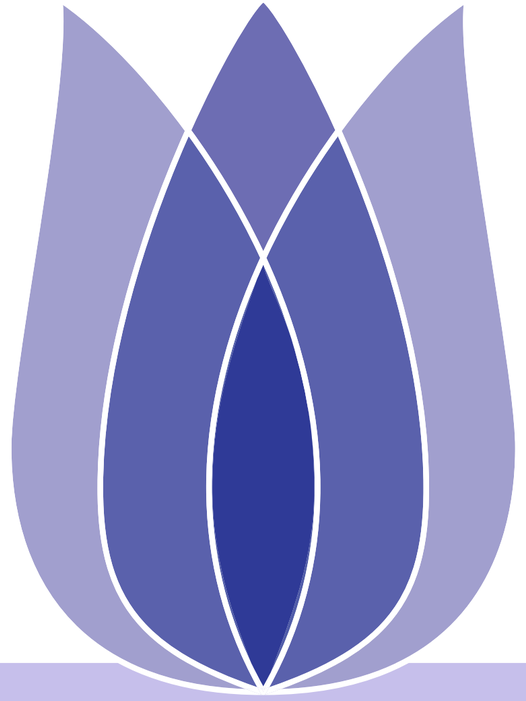
New York City Taxi Trip Duration Prediction

Weiling Deng

Deakin University

(None)



Overview

Problem Definition

- New York City Taxi Trip Duration Prediction

Data ETL

- Data ETL - Data Loading
- Read Data and Print out
- Data Clean and check NA
- Data Discrible

Knowledge Discovery

- Calculate the distance by latitude and longitude
- Data Visualization- Distribution distance
- Distribution Trip Duration and Visualization
- Visualization Distance with Date and Trip Duration

Model Built and Prediction

- Select features and gropby data to train data and test data
- Built LinearRegresion Model and find best parameter to predict
- Visualization the predict data

Conclusion

Problem Definition

New York City Taxi Trip Duration Prediction

Defn

New York City Taxi Trip Duration Prediction aims to identify the outstanding features of the query object.

- We may be interested in the characteristics that make distance trip duration .
- How the relationship between the distance and tripduration, and we use distance to predict trip duration (a query object).

Data ETL

Data ETL - Data Loading

- Data Loading - import data from csv files
  - There are train data and test data:

Train Data

- Name test data as df_train data
- There are 1458644 Rows and 11 columns.
-

Test Data

- Name test data as df_test data
- There are 625134 Rows and 9 columns.

Read Data and Print out


```
#print train data
df_train.head()
```

4]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.765602	
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.731152	
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.710087	
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.012268	40.706718	
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.972923	40.782520	

```
#print test data
df_test.head()
```

7]:

	id	vendor_id	pickup_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	store_and_fwd_flag
0	id3004672	1	2016-06-30 23:59:58	1	-73.988129	40.732029	-73.990173	40.756680	N
1	id3505355	1	2016-06-30 23:59:53	1	-73.964203	40.679993	-73.959808	40.655403	N
2	id1217141	1	2016-06-30 23:59:47	1	-73.997437	40.737583	-73.986160	40.729523	N
3	id2150126	2	2016-06-30 23:59:41	1	-73.956070	40.771900	-73.986427	40.730469	N
4	id1598245	1	2016-06-30 23:59:33	1	-73.970215	40.761475	-73.961510	40.755890	N

Data Clean and check NA


```
#check the NA in train data
df_train.isnull().any()
print(df_train.isnull().any())
```

```
id                  False
vendor_id           False
pickup_datetime     False
dropoff_datetime    False
passenger_count     False
pickup_longitude    False
pickup_latitude     False
dropoff_longitude   False
dropoff_latitude    False
store_and_fwd_flag  False
trip_duration       False
dtype: bool
```

```
#check the NA in test data
df_test.isnull().any()
print(df_test.isnull().any())
```

```
id                  False
vendor_id           False
pickup_datetime     False
passenger_count     False
pickup_longitude    False
pickup_latitude     False
dropoff_longitude   False
dropoff_latitude    False
store_and_fwd_flag  False
dtype: bool
```

- Clean Data and check NA, the result shows that there is no NA

Data Discrible


```
#chaeck the information for train data
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1458644 entries, 0 to 1458643
Data columns (total 11 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   id                  1458644 non-null  object
 1   vendor_id           1458644 non-null  int64
 2   pickup_datetime     1458644 non-null  object
 3   dropoff_datetime    1458644 non-null  object
 4   passenger_count     1458644 non-null  int64
 5   pickup_longitude    1458644 non-null  float64
 6   pickup_latitude     1458644 non-null  float64
 7   dropoff_longitude   1458644 non-null  float64
 8   dropoff_latitude    1458644 non-null  float64
 9   store_and_fwd_flag  1458644 non-null  object
 10  trip_duration       1458644 non-null  int64
dtypes: float64(4), int64(3), object(4)
memory usage: 122.4+ MB
```

```
#check information for test data
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 625134 entries, 0 to 625133
Data columns (total 9 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   id                  625134 non-null  object
 1   vendor_id           625134 non-null  int64
 2   pickup_datetime     625134 non-null  object
 3   passenger_count     625134 non-null  int64
 4   pickup_longitude    625134 non-null  float64
 5   pickup_latitude     625134 non-null  float64
 6   dropoff_longitude   625134 non-null  float64
 7   dropoff_latitude    625134 non-null  float64
 8   store_and_fwd_flag  625134 non-null  object
dtypes: float64(4), int64(2), object(3)
memory usage: 42.9+ MB
```

```
#Train Data decribtion
df_train.describe()
```

2]:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	trip_duration
count	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06	1.458644e+06
mean	1.534950e+00	1.664530e+00	-7.397349e+01	4.075092e+01	-7.397342e+01	4.075180e+01	9.594923e+02
std	4.987772e-01	1.314242e+00	7.090186e-02	3.288119e-02	7.064327e-02	3.589056e-02	5.237432e+03
min	1.000000e+00	0.000000e+00	-1.219333e+02	3.435970e+01	-1.219333e+02	3.218114e+01	1.000000e+00
25%	1.000000e+00	1.000000e+00	-7.399187e+01	4.073735e+01	-7.399133e+01	4.073588e+01	3.970000e+02
50%	2.000000e+00	1.000000e+00	-7.398174e+01	4.075410e+01	-7.397975e+01	4.075452e+01	6.620000e+02
75%	2.000000e+00	2.000000e+00	-7.396733e+01	4.076836e+01	-7.396301e+01	4.076981e+01	1.075000e+03
max	2.000000e+00	9.000000e+00	-6.133553e+01	5.188108e+01	-6.133553e+01	4.392103e+01	3.526282e+06

```
#Test Data decribtion
df_test.describe()
```

3]:

	vendor_id	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
count	625134.000000	625134.000000	625134.000000	625134.000000	625134.000000	625134.000000
mean	1.534884	1.661765	-73.973614	40.750927	-73.973458	40.751816
std	0.498782	1.311293	0.073389	0.029848	0.072565	0.035824
min	1.000000	0.000000	-121.933128	37.389587	-121.933327	36.601322
25%	1.000000	1.000000	-73.991852	40.737392	-73.991318	40.736000
50%	2.000000	1.000000	-73.981743	40.754093	-73.979774	40.754543
75%	2.000000	2.000000	-73.967400	40.768394	-73.963013	40.769852
max	2.000000	9.000000	-69.248917	42.814938	-67.496796	48.857597

Knowledge Discovery

Calculate the distance by latitude and longitude

2.1 Calculate the distance by latitude and longitude

```
#calculated the distance by latitude and longitude
from scipy.spatial import distance
from scipy.spatial.distance import cdist
def haversine(pickup_latitude, pickup_longitude, dropoff_latitude, dropoff_longitude):
    from math import radians, sin, cos, atan2, sqrt
    lat1, long1, lat2, long2 = map(radians, [pickup_latitude, pickup_longitude, dropoff_latitude, dropoff_longitude])
    a = sin((lat1-lat2)/2)**2 + cos(lat1)*cos(lat2)*(sin((long1-long2)/2)**2)
    c = 2 * atan2(sqrt(a), sqrt(1-a))
    return 6371 * c    #  r,km  #
```

```
df_train['distance'] = df_train.apply(lambda row:

                                haversine(row['pickup_latitude'],
                                          row['pickup_longitude'],
                                          row['dropoff_latitude'],
                                          row['dropoff_longitude']), axis=1)
df_test['distance']  = df_test.apply(lambda row:

                               haversine(row['pickup_latitude'],
                                         row['pickup_longitude'],
                                         row['dropoff_latitude'],
                                         row['dropoff_longitude']), axis=1)
```

```
#print train data with distance
df_train
```

5]:

	id	vendor_id	pickup_datetime	dropoff_datetime	passenger_count	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latituc
0	id2875421	2	2016-03-14 17:24:55	2016-03-14 17:32:30	1	-73.982155	40.767937	-73.964630	40.7656(
1	id2377394	1	2016-06-12 00:43:35	2016-06-12 00:54:38	1	-73.980415	40.738564	-73.999481	40.7311!
2	id3858529	2	2016-01-19 11:35:24	2016-01-19 12:10:48	1	-73.979027	40.763939	-74.005333	40.7100!
3	id3504673	2	2016-04-06 19:32:31	2016-04-06 19:39:40	1	-74.010040	40.719971	-74.012268	40.7067
4	id2181028	2	2016-03-26 13:30:55	2016-03-26 13:38:10	1	-73.973053	40.793209	-73.972923	40.7825

Data Visualization- Distribution distance

2.2 Data Visualization

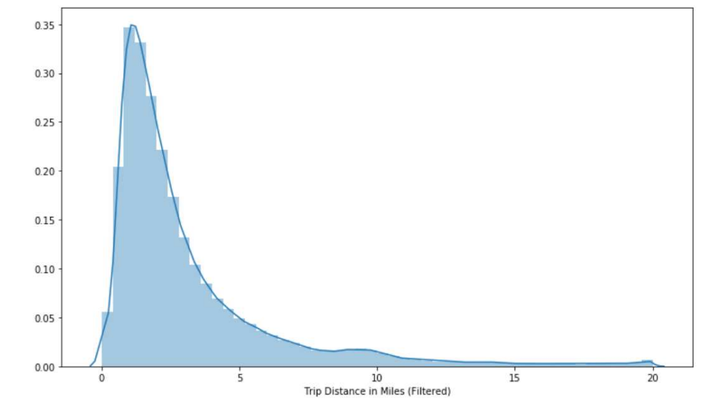
```
#Distribution distance and visualization
df_distance = df_train[df_train['distance'] <= 20].copy()
plt.figure(figsize=(12,7))
sns.distplot(df_distance['distance'].values, axlabel = 'Trip Distance', bins = 50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x18e2e0cf6c8>
```



```
df_distance = df_test[df_test['distance'] <= 20].copy()
plt.figure(figsize=(12,7))
sns.distplot(df_distance['distance'].values, axlabel = 'Trip Distance in Miles (Filtered)', bins = 50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x18e3254f888>
```

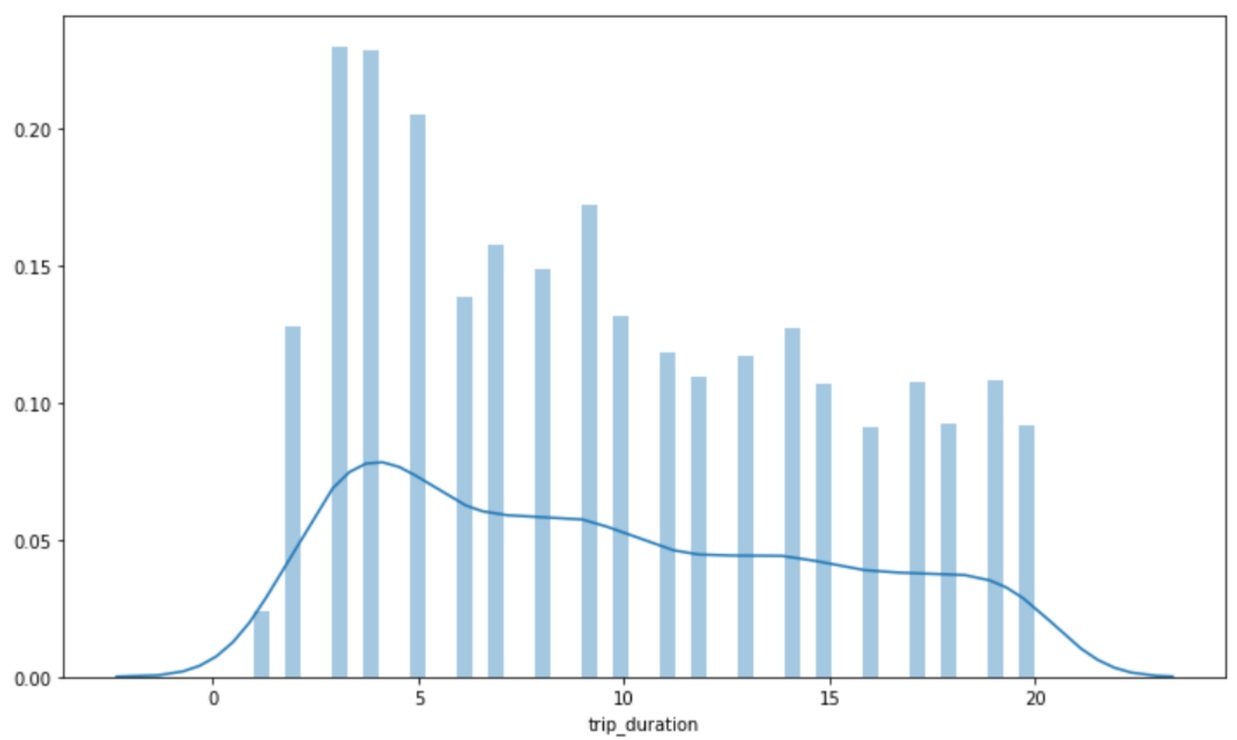


- Distribution distance and visualization the data in train data and test data as well, try to find some features with these data.

Distribution Trip Duration and Visualization

In [22]:
```
df_duration = df_train[df_train['trip_duration'] <=20].copy()
plt.figure(figsize=(12,7))
sns.distplot(df_duration['trip_duration'].values, axlabel = 'trip_duration', bins = 50)
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x18e3278d388>

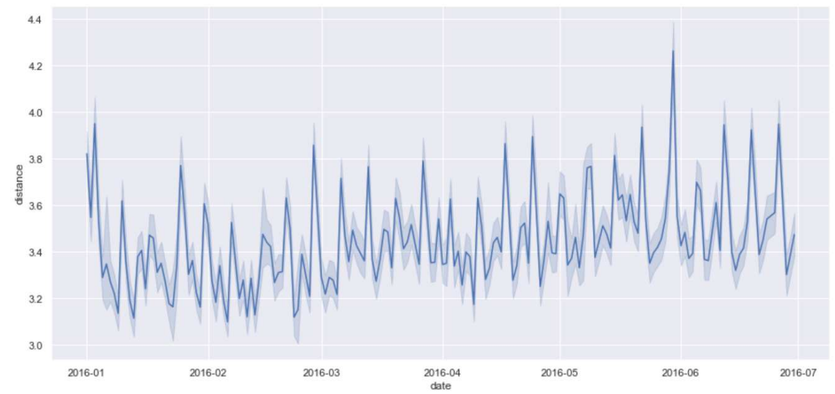


- Distribution trip duration and visualization the data in train data, try to find some features with these data.

Visualization Distance with Date and Trip Duration

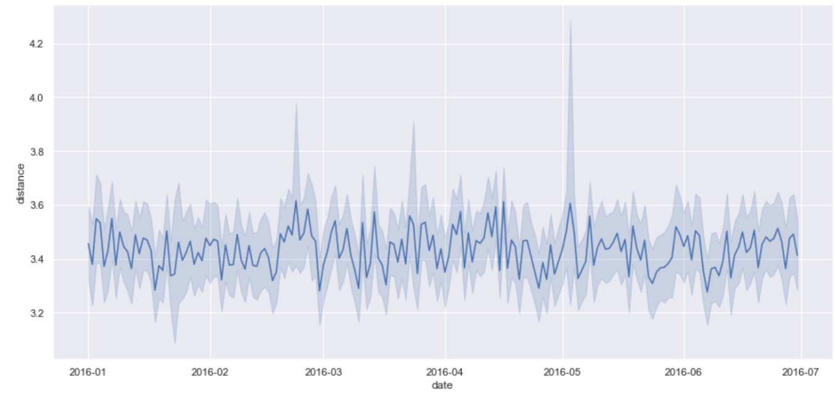

```
sns.set(rc={'figure.figsize':(15, 7)})
sns.lineplot(x="date", y="distance", data=df_train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x18e8d701d88>
```




```
sns.set(rc={'figure.figsize':(15, 7)})
sns.lineplot(x="date", y="distance", data=df_test)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x18e87663e88>
```



- Visualization Distance with Date and Trip Duration, try to find some features with these data.

Model Built and Prediction

Select features and gropby data to train data and test data


```
#Select 'trip_duration' and 'distance' as features
data_train= df_train[['trip_duration','distance']]

data_train
```

]:

	trip_duration	distance
0	455	1.498521
1	663	1.805507
2	2124	6.385098
3	429	1.485498
4	435	1.188588
...	...	...
1458639	778	1.225080
1458640	655	6.049836
1458641	764	7.824606
1458642	373	1.092564
1458643	198	1.134042

1458644 rows × 2 columns

```
#groupby data to train data and test data in the train dataframe
df=data_train
train = df[:150]
test = df[150:180]
train
```

5]:

	trip_duration	distance
0	455	1.498521
1	663	1.805507
2	2124	6.385098
3	429	1.485498
4	435	1.188588
...	...	...
145	972	2.211689
146	318	1.136076
147	625	1.897957
148	859	2.304592
149	141	1.578737

150 rows × 2 columns

- Select ’trip duration’ and ’distance’ as features, groupby data to train data and test data in the train dataframe, set 0 to 150 as train data, 150 to 180 as test data, named as train and test.

Built LinearRegresion Model and find best parameter to predict

3.2 Built LinearRegresion Model and find best parameter to predict

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_log_error

X_train=train['distance'].values.reshape((-1,1))
y_train=train['trip_duration'].values.reshape((-1,1))
X_test=test['distance'].values.reshape((-1,1))
y_test=test['trip_duration'].values.reshape((-1,1))

lr = LinearRegression()
lr.fit(X_train, y_train)
lr_y_predict = lr.predict(y_test)

# checking the rmsle, accuracy
lr_rmsle=np.sqrt(mean_squared_log_error(y_test,lr_y_predict))


print('LinearRegression RMSLE is ',lr_rmsle)
```

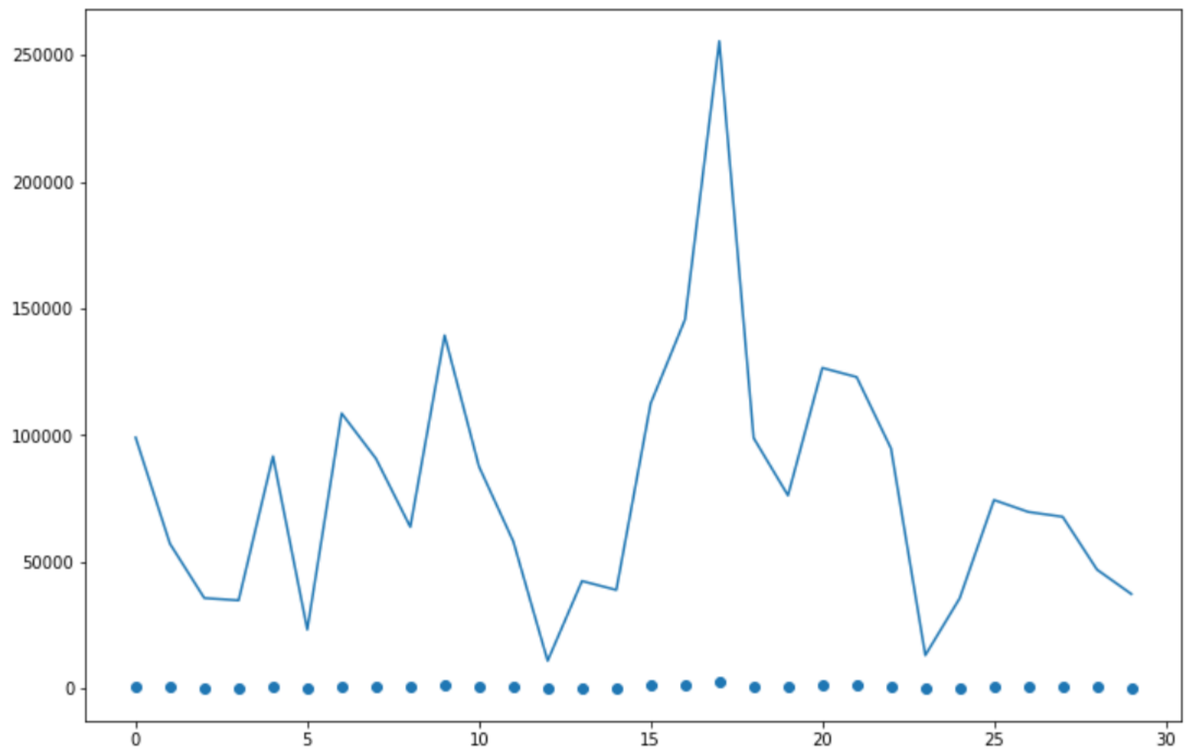
```
LinearRegression RMSLE is  4.616529350404572
```

- Built LinearRegresion Model and find best parameter to predict, we use distance to predict the trip duration and test. The result presents that the LinearRegression RMSLE is 4.616529350404572.

Visualization the predict data


```
#Visualization the predict data
plt.figure(figsize=(12,8))

plt.scatter(np.arange(len(y_test)),y_test,label = 'true value')
plt.plot(np.arange(len(y_test)),lr_y_predict,label = 'predict value')
plt.show()
```



- Visualization the predict data.

Conclusion

Conclusion

- Formalize the problem of *New York City Taxi Trip Duration Prediction*
- Propose to find the data features, select the attruibuates to built the linearregression model, find the best parameter using distance data to predict the trip duration and test.But , from the chart we can see the model is not very good, maybe we an explore other model to improve the accuracy.
- Utilize

Contact Information

Weiling Deng

Business School

Deakin University, Australia

DDENGPP@GMAIL.COM

TEAM FOR FLIP00