Report - Assignment 4

CS5590: Foundations of Machine Learning

Prof. Vineeth N Balasubramanian

Name - Naitik Malav

Roll No. - CS19BTECH11026

Deliverables:

1. I have submitted answers of question 1, 2, 3 and 4 in a hand-written pdf format – Assign4_1_2_3_4.pdf
2. I have submitted two jupyter notebooks – Assign4_5.ipynb and Assign4_6.ipnyb as a solution of question 5th and 6th.
3. And Report.pdf

5th –

a) Logistic regression classifier is trained using gradient descent and cross entropy error function.

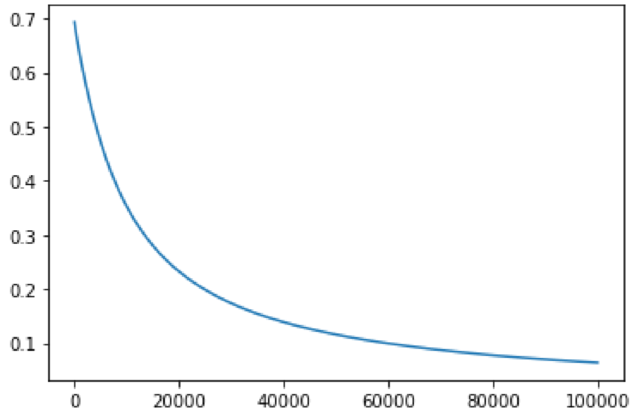
For sample input I have set the weights and bias to zero.
I have set the learning rate = 0.01 and iterations = 100000 and computed the new weights and new bias which are -

```
Weights = [[15.9651542] [ 4.32949164]]
Bias = -11.175304777542943
```

Costs is as follows:

```
Cost after 1 iterations is: 0.6930505532508043
Cost after 10001 iterations is: 0.35214686672757545
Cost after 20001 iterations is: 0.2323229709437589
Cost after 30001 iterations is: 0.1736039027322533
Cost after 40001 iterations is: 0.13887456663575598
Cost after 50001 iterations is: 0.11589771056577576
Cost after 60001 iterations is: 0.09954500104105682
Cost after 70001 iterations is: 0.0872970040762255
Cost after 80001 iterations is: 0.07777130144062172
Cost after 90001 iterations is: 0.07014548494516984
```

With the help of graph one can easily see that cost is decreasing with the no. of iterations.



b)

(i) Weights = [theta1, theta2] = [[1.5], [0.5]] and theta2 = 0.5 and bias = -1
I have passed Weights and bias in logistic regression classifier function. I have set the learning rate as 0.1 and iterations = 100000 for computing new weights and bias which are:

```
Weights = [[31.36347263] [ 7.53262099]]
Bias = -21.19055157257816
```

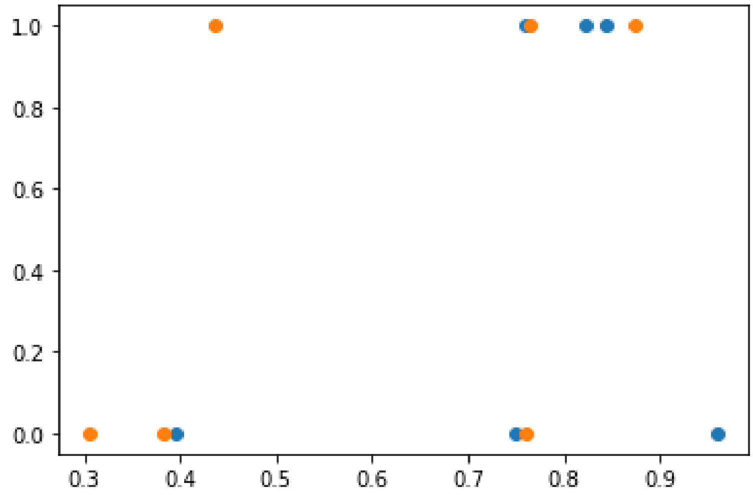
(ii) Here using gradient descent to update theta0, theta1, theta2 for one iteration. And learning rate is set to 0.1.

```
After 1 iteration
New weights are:
 [[1.50535086]
 [0.50196867]]

New Bias is:
 [-1.00316626]
```

(iii) Using the given test set,

```
Accuracy of the model is :  66.6667 %
Precision: 0.600
Recall: 1.000
```

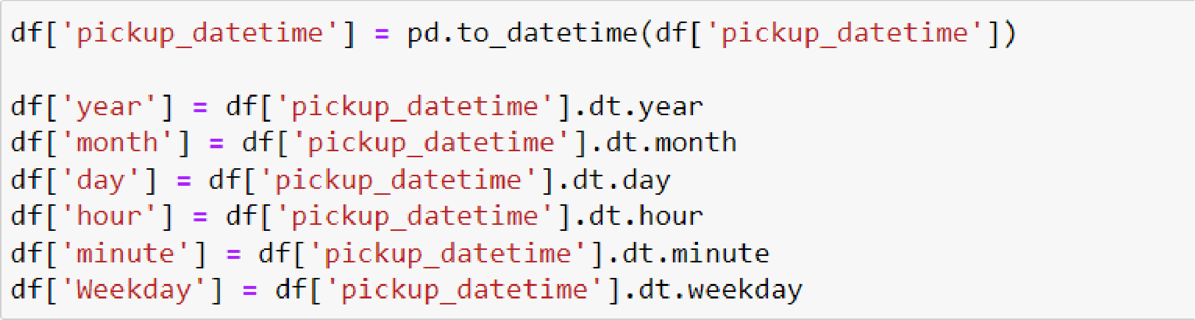
Graph of x1 vs y and x2 vs Y.



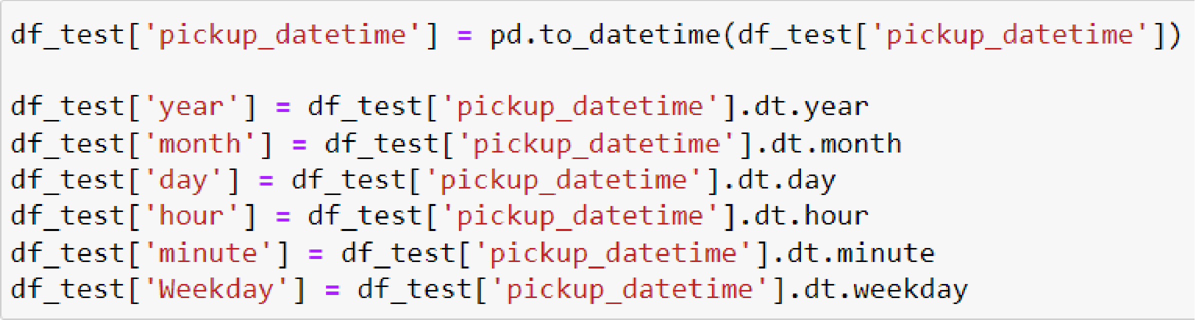
6th –

Training data set is very large so I have selected only first 5,00,000 rows.

Then I have parsed pickup_datetime column into year, month, day, hour, minute, and weekday columns. (for both training and test set)



```
df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'])

df['year'] = df['pickup_datetime'].dt.year
df['month'] = df['pickup_datetime'].dt.month
df['day'] = df['pickup_datetime'].dt.day
df['hour'] = df['pickup_datetime'].dt.hour
df['minute'] = df['pickup_datetime'].dt.minute
df['Weekday'] = df['pickup_datetime'].dt.weekday
```



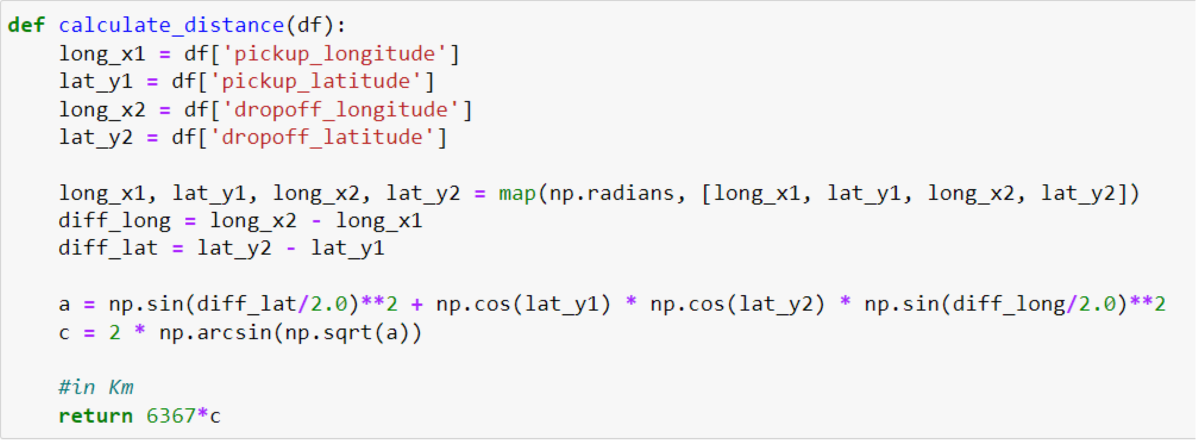
```
df_test['pickup_datetime'] = pd.to_datetime(df_test['pickup_datetime'])

df_test['year'] = df_test['pickup_datetime'].dt.year
df_test['month'] = df_test['pickup_datetime'].dt.month
df_test['day'] = df_test['pickup_datetime'].dt.day
df_test['hour'] = df_test['pickup_datetime'].dt.hour
df_test['minute'] = df_test['pickup_datetime'].dt.minute
df_test['Weekday'] = df_test['pickup_datetime'].dt.weekday
```

After that I have dropped those rows which are having NaN values in training dataset. So this is the no. of left rows and columns = `(499995, 14)`

Then normalising the training sets column. If the absolute difference between column value and it’s mean value then dropping those values. Hence no. of left rows are `(489696, 14)`

Then dropping keys and pickup_datetime.

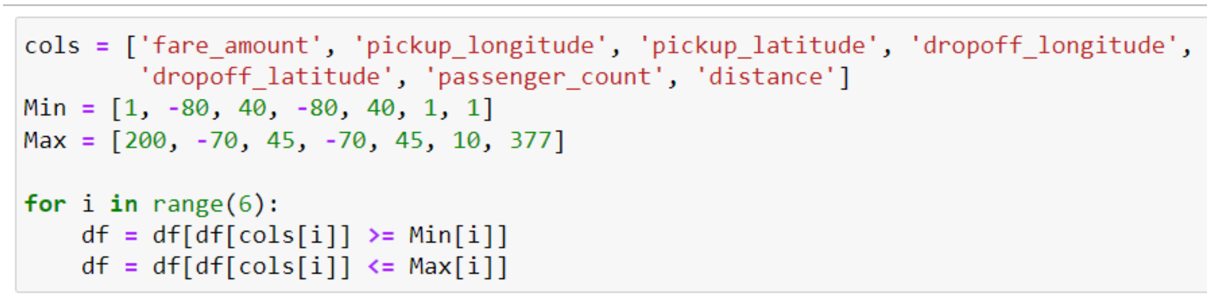
I have also added a distance column in both training and test dataset. I have calculated the haversine distance with the help of pickup latitude, longitude and drop off latitude longitude.
I have used the formula given on this link.



```
def calculate_distance(df):
    long_x1 = df['pickup_longitude']
    lat_y1 = df['pickup_latitude']
    long_x2 = df['dropoff_longitude']
    lat_y2 = df['dropoff_latitude']

    long_x1, lat_y1, long_x2, lat_y2 = map(np.radians, [long_x1, lat_y1, long_x2, lat_y2])
    diff_long = long_x2 - long_x1
    diff_lat = lat_y2 - lat_y1

    a = np.sin(diff_lat/2.0)**2 + np.cos(lat_y1) * np.cos(lat_y2) * np.sin(diff_long/2.0)**2
    c = 2 * np.arcsin(np.sqrt(a))

    #in Km
    return 6367*c
```

Then I have used random forest regressor with n_estimators=1000, and max_depth=8, but I got RMSE score more than 5. So, to reduce it we have to remove the outliers.

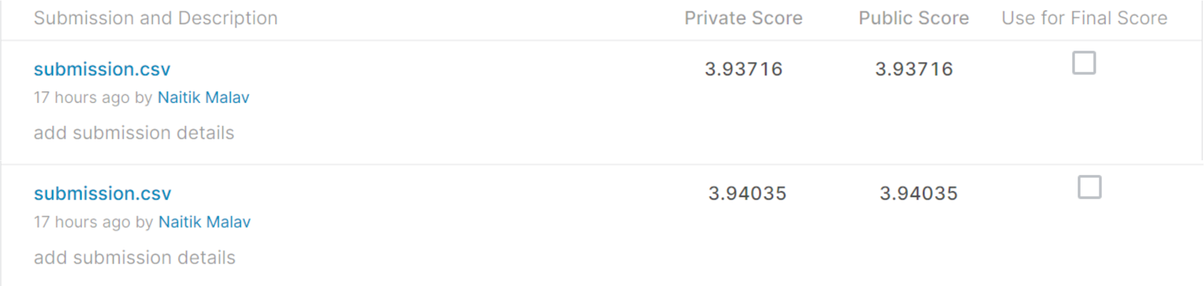
So, I have selected the below ranges for the outliers for the cols.



```
cols = ['fare_amount', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
        'dropoff_latitude', 'passenger_count', 'distance']
Min = [1, -80, 40, -80, 40, 1, 1]
Max = [200, -70, 45, -70, 45, 10, 377]

for i in range(6):
    df = df[df[cols[i]] >= Min[i]]
    df = df[df[cols[i]] <= Max[i]]
```

Now no. of rows are `(487802, 13)`

Now, this time RMSE score is, **3.94035**
(with n_estimators=500, oob_score=true, max_depth=8)
Now this time RMSE score is, **3.93716**
(with n_estimators=1000, oob_score=true, max_depth=8)



Submission and Description	Private Score	Public Score	Use for Final Score
submission.csv 17 hours ago by Naitik Malav add submission details	3.93716	3.93716	□
submission.csv 17 hours ago by Naitik Malav add submission details	3.94035	3.94035	□

(Can't make these both as final score, it's saying it's not allowed after deadline, so attaching it here).