Linear Regression Analysis and Data Analysis for Taxi Fares Prediction in New York City

Hosea Verico Dwi Kristianto | sayahoseaverico@gmail.com

*picture by **wirestock** from **freepik**.*

Hi everyone...!! my name is Hosea Verico Dwi Kristianto and today I will share to you guys about my analysis result about taxi fares in the New York City. The dataset I'm working on come from New York City Taxi and Limousine Commission (TLC) transaction. I will get into the data and search to some of its value that could be responsible for the taxi fares(price). Here I'm using **Python** programming language to process and gaining the insight from the dataset along with some data analyzing library such as matplotlib, numpy, pandas, sci-kit learn.

Lets jump into the analysis....!!!

Analysis

Data Preparation

Data from CSV file "2017_Yellow_Taxi_Trip_Data.CSV" are consist of 22.000+ rows of data and has 18 column in it.

```
# Load dataset into dataframe
df0=pd.read_csv("2017_Yellow_Taxi_Trip_Data.csv")
```

```
Dataframe rows :  22699
Dataframe collumn :  18

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22699 entries, 0 to 22698
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             22699 non-null  int64
 1   VendorID               22699 non-null  int64
 2   tpep_pickup_datetime   22699 non-null  object
 3   tpep_dropoff_datetime  22699 non-null  object
 4   passenger_count        22699 non-null  int64
 5   trip_distance          22699 non-null  float64
 6   RatecodeID             22699 non-null  int64
 7   store_and_fwd_flag     22699 non-null  object
 8   PULocationID           22699 non-null  int64
 9   DOLocationID           22699 non-null  int64
 10  payment_type           22699 non-null  int64
 11  fare_amount            22699 non-null  float64
 12  extra                  22699 non-null  float64
 13  mta_tax                22699 non-null  float64
 14  tip_amount             22699 non-null  float64
 15  tolls_amount           22699 non-null  float64
 16  improvement_surcharge  22699 non-null  float64
 17  total_amount           22699 non-null  float64
```

From the dataset I've imported I realized I cant use all of the available column and need to twist some of its value for normalization purpose. I decided to just use the duration and distance of the trip also I think that the amount of the passenger would also contribute to the price of the transaction. The duration of a transaction are the mean of the substracted drop time by pickup time, while the distance are the mean of trip distance from pickup and drop out location.

```python
# 1. Create a mean_distance column that is a copy of the pickup_dropoff helper column
### YOUR CODE HERE ###
df0['mean_distance'] = df0['pickup_dropoff']

# 2. Map `grouped_dict` to the `mean_distance` column
### YOUR CODE HERE ###
df0['mean_distance'] = df0['mean_distance'].map(grouped_dictionary)

# Confirm that it worked
### YOUR CODE HERE ###
df0[['mean_distance']].head()
```

	mean_distance
0	3.521667
1	3.108889
2	0.881429
3	3.700000
4	4.435000

```python
# Create a dictionary where keys are unique pickup_dropoffs and values are
# mean trip duration for all trips with those pickup_dropoff combos
### YOUR CODE HERE ###
grouped_trip_duration = df0.groupby('pickup_dropoff').mean(numeric_only = True)[["duration"]]

trip_duration_dict = grouped_trip_duration.to_dict()
trip_duration_dict = trip_duration_dict['duration']

##create a new column and map the grouped mean value
df0['mean_duration'] = df0['pickup_dropoff']
df0['mean_duration'] = df0['mean_duration'].map(trip_duration_dict)

# Confirm that it worked
### YOUR CODE HERE ###
df0[["mean_duration"]].head()
```

	mean_duration
0	22.847222
1	24.470370
2	7.250000
3	30.250000
4	14.616667

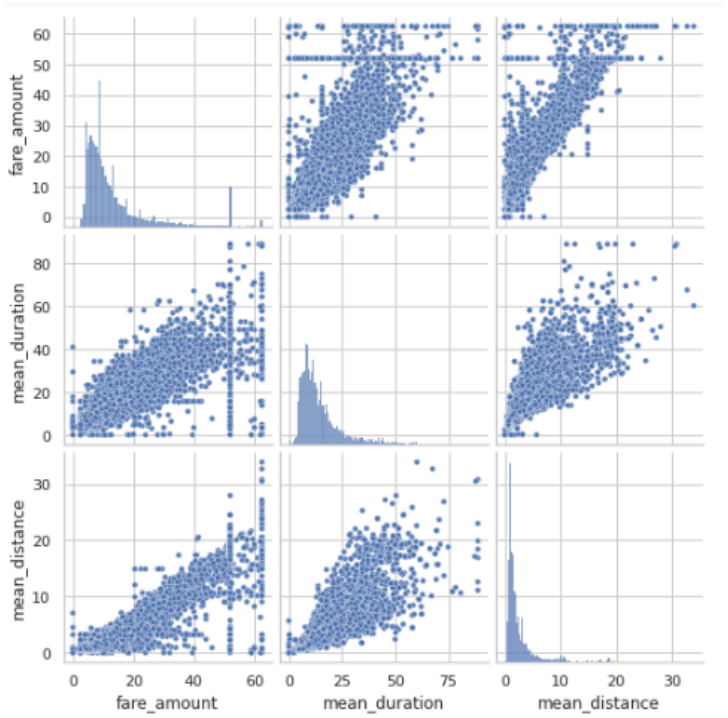
After doing some twist and change to the data I can use the mean distance and duration of the transaction for regression analysis. I also used the number of passenger and add a rush hour value to dataset. After that I will isolate all that variable for the next step.

```
# Dependant variable : fare_amount
# Independent variable : mean_distance, mean_duration, rush_hour, VendorID, passenger_count

df1 = df0.copy()
listess = ["VendorID", "passenger_count","mean_distance", "mean_duration", "rush_hour","fare_amount"]

df1 = df1.drop(['Unnamed: 0', 'tpep_dropoff_datetime', 'tpep_pickup_datetime',
                'trip_distance', 'RatecodeID', 'store_and_fwd_flag', 'PULocationID', 'DOLocationID',
                'payment_type', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount', 'improvement_surcharge',
                'total_amount', 'tpep_dropoff_datetime', 'tpep_pickup_datetime', 'duration',
                'pickup_dropoff', 'day', 'month'
                ], axis=1)
df1.head()
```

	VendorID	passenger_count	fare_amount	mean_distance	mean_duration	rush_hour
0	2	6	13.0	3.521667	22.847222	0
1	1	1	16.0	3.108889	24.470370	0
2	1	1	6.5	0.881429	7.250000	1
3	2	1	20.5	3.700000	30.250000	0
4	2	1	16.5	4.435000	14.616667	0

Regression Analysis

On this step I will **Analyzing** the linear regression of the values. First I will go and check some of the assumption I can get from the dependent and independent variable, especially the distance and duration of the transaction to the fare amount

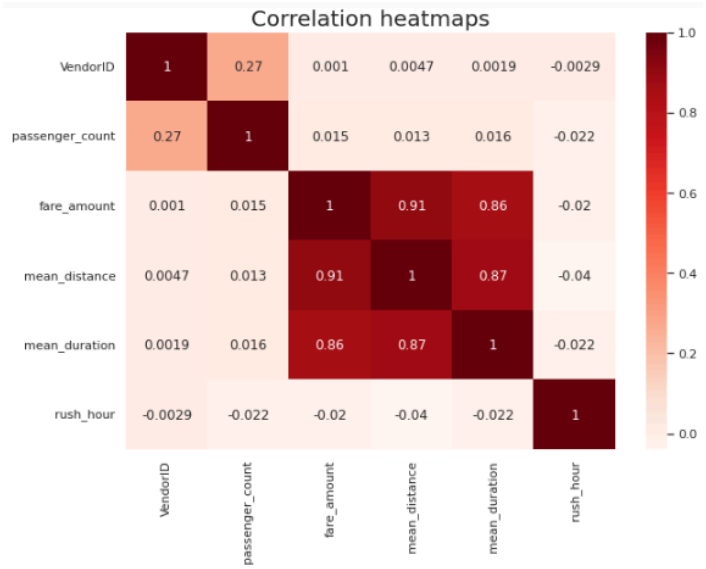


mean_duration, mean_distance to fare_amount pair plot

from the pairplot shown I can conclude that the mean_distance and the mean duration are quite linear to the fair amount of the transaction shown by the [mean_distance, fare_amount] plot and [mean_duration,

fare_amount] plot are following a linear distribution so that we can confirm the linearity assumption for the dataset.

Next I will check the no multi collinearity assumption for the both mean_distance and the mean_duration, it shown by [mean_distance, mean_duration], [mean_duration, mean_distance] plot values are not quite linear to each other and more spread out. The no multi collinearity assumption are important to check because the violation on this assumption could fraud the regression analysis and creating bias due to the highly correlated value of the models. To further check the no multi collinearity assumption we can cross checking the assumption by using pearson table and check if there are variable that have way to high correlation matrix number, its should be bellow 0.80.



With the heatmap shown that the mean_distance and the mean_duration are quite correlated so that I cant just use them for modeling and need to preprocessing them before using them.

Data Preprocessing

As I've mention before I need to preprocess the data again so that we can properly use them here I use standardization.

```
# Standardize the X variables
### YOUR CODE HERE ###
scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_train_scaled
```

```
array([[ 0.89575785, -0.49880314, -0.03750269,  0.01271026, -0.65545204],
       [ 0.89575785, -0.49880314,  0.45538104, -0.03750208,  1.52566463],
       [ 0.89575785,  0.28284738, -0.42829783,  0.03412234, -0.65545204],
       ...,
       [ 0.89575785,  0.28284738, -0.51433299, -0.5059124 ,  1.52566463],
       [ 0.89575785,  0.28284738, -0.23134537,  0.39237496, -0.65545204],
       [ 0.89575785, -0.49880314,  1.81879464,  0.76643958, -0.65545204]])
```

Modeling

After done checking the assumption and preprocessing the dataset variable I will start modeling the linear regression.

```
# Fit your model to the training data
### YOUR CODE HERE ###
LineaRed = LinearRegression()
LineaRed.fit(X_train_scaled, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

After the model is ready I will test them to the train data and pull out the evaluation matrix from it.

```
# Evaluate the model performance on the training data
### YOUR CODE HERE ###
lr_sc =  LineaRed.score(X_train_scaled, y_train)
print("Linear Regression Evaluation Score : ", lr_sc)
y_train_predict = LineaRed.predict(X_train_scaled)
print("Linear Regression R2 score : ", r2_score(y_train, y_train_predict))
print("Linear Regression MAE : ", mean_absolute_error(y_train, y_train_predict))
print("Linear Regression MSE : ", mean_squared_error(y_train, y_train_predict))
print("Linear Regression RMSE : ", np.sqrt(mean_squared_error(y_train, y_train_predict)))
```

```
Linear Regression Evaluation Score :  0.8488017574563115
Linear Regression R2 score :  0.8488017574563115
Linear Regression MAE :  2.181924595628648
Linear Regression MSE :  17.088312217818014
Linear Regression RMSE :  4.13380118266687
```

I will compare the evaluation matrix between the train data and the test data for validation.

```
# Scale the X_test data
### YOUR CODE HERE ###
X_test_scaled = scaler.transform(X_test)
```

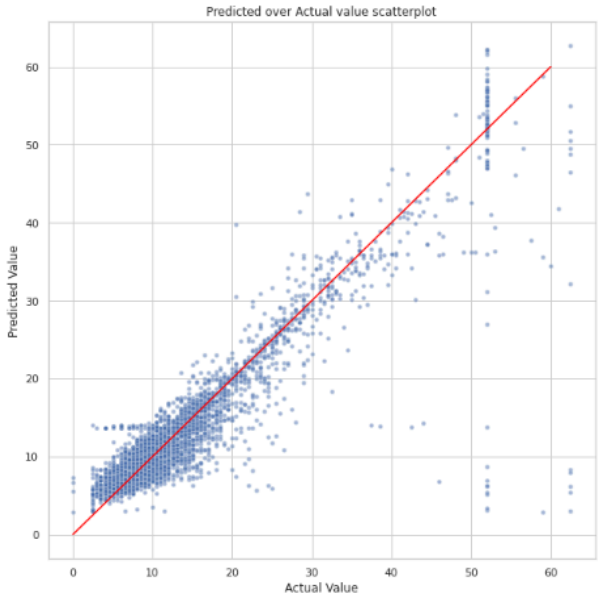
```
# Evaluate the model performance on the testing data
### YOUR CODE HERE ###
lr_sc =  LineaRed.score(X_test_scaled, y_test)
print("Linear Regression Evaluation Score : ", lr_sc)
y_test_predict = LineaRed.predict(X_test_scaled)
print("Linear Regression R2 score : ", r2_score(y_test, y_test_predict))
print("Linear Regression MAE : ", mean_absolute_error(y_test, y_test_predict))
print("Linear Regression MSE : ", mean_squared_error(y_test, y_test_predict))
print("Linear Regression RMSE : ", np.sqrt(mean_squared_error(y_test, y_test_predict)))
```

```
Linear Regression Evaluation Score :  0.8304785329674021
Linear Regression R2 score :  0.8304785329674023
Linear Regression MAE :  2.124781414213245
Linear Regression MSE :  17.521791212694772
Linear Regression RMSE :  4.185903870455552
```

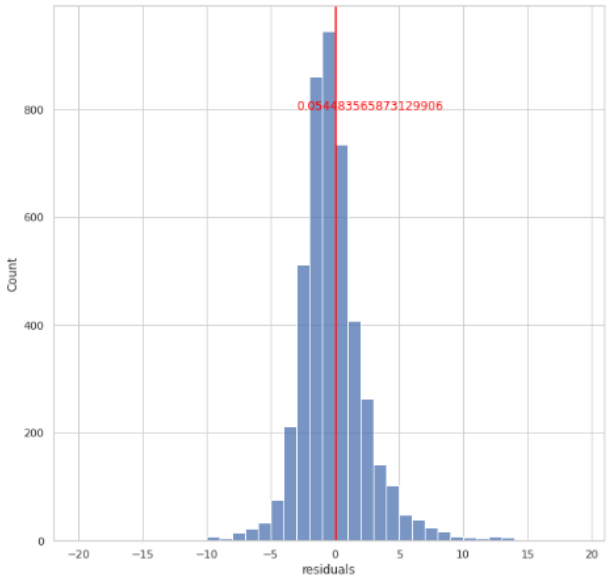
The code shows that the model work on similar performance on both of the dataset split and did not show any major sign of overfitting or underfitting.

Results

After working on the regression analysis and the regression model I can finally conclude the result of the fare amount for Taxi in New York City and try to conclude the next assumption over our dataset that is the **Normality Assumption** where this assumption can only decided after the model are complete. I will visualize the value of the fare amount from the dataset with the fare amount I get from the model.



The scatter plot shows that the actual and predicted values are follows a straight line it shows that the model are working well predicting the actual values of fare amount. Next I`m going to check the next assumption for the dataset with the **Normality Assumption**. This assumption state that a dataset model residue distribution are following normal distribution (bell shaped distribution).



The visualization shows the bell shaped distribution for residuals of model predicted value over the actual value, it confirm the **Normality Assumption** on our dataset.

Conclusion

With all the steps and math done the conclusion are clear that the fare amount(price) of a taxi in New York City are corelated by the distance of the trip, the duration of the trip, the amount of the passenger, and if the trip are on a rush hour or not. Its because when these variable are feed to the linear regression model the predicted value are following a linear line with the actually value of fare amount. So to minimize the fare amount of the taxi when we are in New York City we can try to reduce some of these variable .

That’s all I can show you guys in this occasion, hope my analysis could help you with the fare amount when using taxi in New York City. Good byeeeee.....
