qwucommpi

May 19, 2025

1 Food Delivery Time Prediction

[1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
```

[2]:
```
data =  pd.read_csv("deliverytime.txt")
data.head(10)
```

[2]:

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	\
0	4607	INDORES13DEL02	37	4.9	
1	B379	BANGRES18DEL02	34	4.5	
2	5D6D	BANGRES19DEL01	23	4.4	
3	7A6A	COIMBRES13DEL02	38	4.7	
4	70A2	CHENRES12DEL01	32	4.6	
5	9BB4	HYDRES09DEL03	22	4.8	
6	95B4	RANCHIRES15DEL01	33	4.7	
7	9EB2	MYSRES15DEL02	35	4.6	
8	1102	HYDRES05DEL02	22	4.8	
9	CDCD	DEHRES17DEL01	36	4.2	

	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	\
0	22.745049	75.892471	22.765049	
1	12.913041	77.683237	13.043041	
2	12.914264	77.678400	12.924264	
3	11.003669	76.976494	11.053669	
4	12.972793	80.249982	13.012793	
5	17.431668	78.408321	17.461668	
6	23.369746	85.339820	23.479746	
7	12.352058	76.606650	12.482058	
8	17.433809	78.386744	17.563809	
9	30.327968	78.046106	30.397968	

	Delivery_location_longitude	Type_of_order	Type_of_vehicle	Time_taken(min)
0	75.912471	Snack	motorcycle	24
1	77.813237	Snack	scooter	33
2	77.688400	Drinks	motorcycle	26

```
3                    77.026494       Buffet      motorcycle                21
4                    80.289982        Snack         scooter                30
5                    78.438321       Buffet      motorcycle                26
6                    85.449820         Meal         scooter                40
7                    76.736650         Meal      motorcycle                32
8                    78.516744       Buffet      motorcycle                34
9                    78.116106        Snack      motorcycle                46
```

[3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ID                           45593 non-null  object 
 1   Delivery_person_ID           45593 non-null  object 
 2   Delivery_person_Age          45593 non-null  int64  
 3   Delivery_person_Ratings      45593 non-null  float64
 4   Restaurant_latitude          45593 non-null  float64
 5   Restaurant_longitude         45593 non-null  float64
 6   Delivery_location_latitude   45593 non-null  float64
 7   Delivery_location_longitude  45593 non-null  float64
 8   Type_of_order                45593 non-null  object 
 9   Type_of_vehicle              45593 non-null  object 
 10  Time_taken(min)              45593 non-null  int64  
dtypes: float64(5), int64(2), object(4)
memory usage: 3.8+ MB
```

[4]:
```
data.isnull().sum()
```

[4]:
```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Type_of_order                  0
Type_of_vehicle                0
Time_taken(min)                0
dtype: int64
```

2 Calculating Distance Between Two Latitudes and Longitudes

[5]:
```
# Setting the Earth's radius (in kilometers)
R = 6371

# Converting degrees to radians
def deg_to_rad(degrees):
    return degrees * (np.pi/180)

# Function to calculate the distance between two points using the haversine␣
 ↪formula
def distcalculate(lat1, lon1, lat2, lon2):
    d_lat = deg_to_rad(lat2-lat1)
    d_lon = deg_to_rad(lon2-lon1)
    a = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1)) * np.
 ↪cos(deg_to_rad(lat2)) * np.sin(d_lon/2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return R * c

# Calculate the distance between each pair of points
data['distance'] = np.nan

for i in range(len(data)):
    data.loc[i, 'distance'] = distcalculate(data.loc[i, 'Restaurant_latitude'],
                                            data.loc[i, 'Restaurant_longitude'],
                                            data.loc[i,␣
 ↪'Delivery_location_latitude'],
                                            data.loc[i,␣
 ↪'Delivery_location_longitude'])
```

[6]:
```
data.head(10)
```

[6]:

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	\
0	4607	INDORES13DEL02	37	4.9	
1	B379	BANGRES18DEL02	34	4.5	
2	5D6D	BANGRES19DEL01	23	4.4	
3	7A6A	COIMBRES13DEL02	38	4.7	
4	70A2	CHENRES12DEL01	32	4.6	
5	9BB4	HYDRES09DEL03	22	4.8	
6	95B4	RANCHIRES15DEL01	33	4.7	
7	9EB2	MYSRES15DEL02	35	4.6	
8	1102	HYDRES05DEL02	22	4.8	
9	CDCD	DEHRES17DEL01	36	4.2	

	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	\
0	22.745049	75.892471	22.765049	
1	12.913041	77.683237	13.043041	

```
2            12.914264            77.678400                  12.924264
3            11.003669            76.976494                  11.053669
4            12.972793            80.249982                  13.012793
5            17.431668            78.408321                  17.461668
6            23.369746            85.339820                  23.479746
7            12.352058            76.606650                  12.482058
8            17.433809            78.386744                  17.563809
9            30.327968            78.046106                  30.397968

   Delivery_location_longitude Type_of_order Type_of_vehicle  Time_taken(min)  \
0                    75.912471        Snack      motorcycle               24
1                    77.813237        Snack         scooter               33
2                    77.688400       Drinks      motorcycle               26
3                    77.026494       Buffet      motorcycle               21
4                    80.289982        Snack         scooter               30
5                    78.438321       Buffet      motorcycle               26
6                    85.449820         Meal         scooter               40
7                    76.736650         Meal      motorcycle               32
8                    78.516744       Buffet      motorcycle               34
9                    78.116106        Snack      motorcycle               46

    distance
0   3.025149
1  20.183530
2   1.552758
3   7.790401
4   6.210138
5   4.610365
6  16.600361
7  20.205253
8  19.975520
9  10.280582
```

[7]:
```
figure = px.scatter(data_frame = data,
                    x='distance',
                    y='Time_taken(min)',
                    size = 'Time_taken(min)',
                    trendline = 'ols',
                    title = 'Relationship Between Distance and Time Taken')
figure.show()
```

2.0.1 There is a consistent relationship between the time taken and the distance travelled to deliver the food. It means that most delivery partners deliver food within 25-30 minutes, regardless of distance.

2.1 Q. Relationship between the time taken to deliver the food and the age of the delivery partner

[8]:
```
figure = px.scatter(data_frame = data,
                    x='Delivery_person_Age',
                    y='Time_taken(min)',
                    size='Time_taken(min)',
                    color='distance',
                    trendline='ols',
                    title='Relationship Between Time Taken and Age')
figure.show()
```

There is a linear relationship between the time taken to deliver the food and the age of the delivery partner. It means young delivery partners take less time to deliver the food compared to the elder partners.

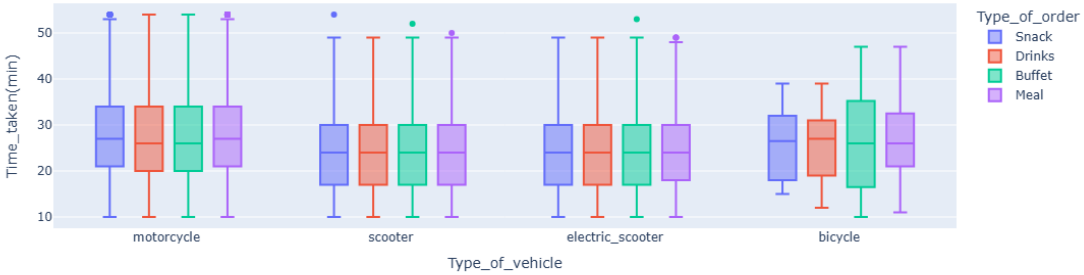
2.1.1 Q. Finding the relationship between the time taken to deliver the food and the ratings of the delivery partner

[9]:
```
figure = px.scatter(data_frame = data,
                     x="Delivery_person_Ratings",
                     y="Time_taken(min)",
                     size="Time_taken(min)",
                     color = "distance",
                     trendline="ols",
                     title = "Relationship Between Time Taken and Ratings")
figure.show()
```

There is an inverse linear relationship between the time taken to deliver the food and the ratings of the delivery partner. It means delivery partners with higher ratings take less time to deliver the food compared to partners with low ratings.

2.1.2 Q. Now if the type of food ordered by the customer and the type of vehicle used by the delivery partner affects the delivery time or not:

[10]:
```
fig = px.box(data,
             x='Type_of_vehicle',
             y='Time_taken(min)',
             color = 'Type_of_order')
fig.show()
```



2.1.3 So there is not much difference between the time taken by delivery partners depending on the vehicle they are driving and the type of food they are delivering.

3 Food Delivery Time Prediction Model

Now training a Machine Learning model using an LSTM neural network model for the task of food delivery time prediction:

[11]:
```
# splitting the data
from sklearn.model_selection import train_test_split
```

[13]:
```
x = np.array(data[["Delivery_person_Age",
                   "Delivery_person_Ratings",
                   "distance"]])
y = np.array(data[["Time_taken(min)"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.10,
                                                random_state=42)
```

[21]:
```
# creating LSTM (Long-Short-Term-Memory) neural network model

from keras.models import Sequential
from keras.layers import Dense, LSTM
```

[23]:
```
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (xtrain.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 3, 128)	66,560
lstm_2 (LSTM)	(None, 64)	49,408
dense (Dense)	(None, 25)	1,625
dense_1 (Dense)	(None, 1)	26

Total params: 117,619 (459.45 KB)

Trainable params: 117,619 (459.45 KB)

Non-trainable params: 0 (0.00 B)

[25]:
```
# training the model
model.compile(optimizer = 'adam',loss = 'mean_squared_error')
model.fit(xtrain,ytrain,batch_size=1,epochs=9)
```

```
Epoch 1/9
41033/41033             141s 3ms/step
- loss: 76.1485
Epoch 2/9
41033/41033             139s 3ms/step
- loss: 64.9759
Epoch 3/9
41033/41033             142s 3ms/step
- loss: 62.2258
Epoch 4/9
41033/41033             140s 3ms/step
- loss: 61.0707
Epoch 5/9
41033/41033             141s 3ms/step
- loss: 60.1171
Epoch 6/9
41033/41033             143s 3ms/step
- loss: 60.4330
Epoch 7/9
41033/41033             140s 3ms/step
- loss: 58.7293
Epoch 8/9
41033/41033             142s 3ms/step
```

```
- loss: 58.9499
Epoch 9/9
41033/41033             143s 3ms/step
- loss: 59.4311
```

[25]:
```
<keras.src.callbacks.history.History at 0x1a0576c1c70>
```

Now let's test the performance of our model by giving inputs to predict the food delivery time:

[34]:
```
print("Food Delivery Time Prediction")
a = int(input("Age of Delivery Partner(Year): "))
b = float(input("Ratings of Previous Deliveries(1-5): "))
c = int(input("Total Distance(km): "))
features = np.array([[a, b, c]])
print("Predicted Delivery Time in Minutes = ", model.predict(features))
```

```
Food Delivery Time Prediction

Age of Delivery Partner(Year):  40
Ratings of Previous Deliveries(1-5):  4
Total Distance(km):  5

1/1             0s 42ms/step
Predicted Delivery Time in Minutes =  [[34.43443]]
```

Conclusion Thus we Can see the predictiove model is showing the Delivery Time after we are giving the input

[ ]: