Food Delivery Time Prediction

- The dataset you are given here is a cleaned version of the original dataset submitted by Gaurav Malik on Kaggle.

https://www.kaggle.com/gauravmalik26

- Below are all the features in the dataset:

1. ID: order ID number
2. Delivery_person_ID: ID number of the delivery partner
3. Delivery_person_Age: Age of the delivery partner
4. Delivery_person_Ratings: ratings of the delivery partner based on past deliveries
5. Restaurant_latitude: The latitude of the restaurant
6. Restaurant_longitude: The longitude of the restaurant
7. Delivery_location_latitude: The latitude of the delivery location
8. Delivery_location_longitude: The longitude of the delivery location
9. Type_of_order: The type of meal ordered by the customer
10. Type_of_vehicle: The type of vehicle delivery partner rides
11. Time_taken(min): The time taken by the delivery partner to complete the order

- You are required to predict the delivery time based on the distance covered by the delivery partner to deliver the order.

Importing Libraries

In [ ]:

```
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
```

Loading Data

In [ ]:

```
df = pd.read_csv("./Delivery time/deliverytime.csv")
df
```

Out[ ]:

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delive
0	4607	INDORES13DEL02	37	4.9	22.745049	75.892471	
1	B379	BANGRES18DEL02	34	4.5	12.913041	77.683237	
2	5D6D	BANGRES19DEL01	23	4.4	12.914264	77.678400	
3	7A6A	COIMBRES13DEL02	38	4.7	11.003669	76.976494	
4	70A2	CHENRES12DEL01	32	4.6	12.972793	80.249982	
...	...	...	...	...	...	...	
45588	7C09	JAPRES04DEL01	30	4.8	26.902328	75.794257	
45589	D641	AGRRES16DEL01	21	4.6	0.000000	0.000000	
45590	4F8D	CHENRES08DEL03	30	4.9	13.022394	80.242439	
45591	5EEE	COIMBRES11DEL01	20	4.7	11.001753	76.986241	
45592	5FB2	RANCHIRES09DEL02	23	4.9	23.351058	85.325731	

45593 rows × 11 columns

Getting Insights of Data

In [ ]:

```
df.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
```

In [ ]:

```python
df.isnull().sum()
```

Out[ ]:

```
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
```

- There is no Null value presernt in the data

- Using Haversine formula to calculate the distance between two location based on latitude and longitude

In [ ]:

```python
# Set the earth's radius (in kilometers)
R = 6371
```

In [ ]:

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

In [ ]:

```python
# 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
```

In [ ]:

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

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

- added a column distance (between resturant and delivery location) in the dataset

In [ ]:

```python
print(df.head())
```

```
     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   

   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   

   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   

    distance  
0   3.025149  
1  20.183530  
2   1.552758  
3   7.790401  
4   6.210138  
```

DATA Visualization

In [ ]:

```python
figure = px.scatter(data_frame = df, 
                    x="distance",
                    y="Time_taken(min)", 
                    size="Time_taken(min)", 
                    trendline="ols", 
                    title = "Relationship Between Distance and Time Taken")
figure.show()
```

- There is a constant relation between the time taken and the distance travelled to deliver
- According to the graph, most delivery partners delivers food within 25 to 30 minutes

In [ ]:

```python
figure = px.scatter(data_frame = df, 
                    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.

In [ ]:

```python
figure = px.scatter(data_frame = df, 
                    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.

In [ ]:

```python
fig = px.box(df, 
             x="Type_of_vehicle",
             y="Time_taken(min)", 
             color="Type_of_order")
fig.show()
```

- 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.

Concluding:

- The features that contribute most to the food delivery time based on our analysis are:

1. age of the delivery partner
2. rating of the delivery partner
3. distance b/w the resturent and the delivery location

Food Delivery Time Prediction Model

In [ ]:

```
#splitting data
from sklearn.model_selection import train_test_split
x = np.array(df[["Delivery_person_Age",
                   "Delivery_person_Ratings",
                   "distance"]])
y = np.array(df[["Time_taken(min)"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.10,
                                                random_state=42)
```

In [ ]:

```
# creating the LSTM neural network model
from keras.models import Sequential
from keras.layers import Dense, LSTM
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()
```

```
c:\Users\ASUS\anaconda3\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                      ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm (LSTM)                       │ (None, 3, 128)         │        66,560 │
├───────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_1 (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)
```

In [ ]:

```
# training the model
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=3, epochs=9)
```

```
Epoch 1/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 38s 3ms/step - loss: 64.0704
Epoch 2/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 34s 2ms/step - loss: 62.0130
Epoch 3/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 34s 3ms/step - loss: 60.2492
Epoch 4/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 39s 3ms/step - loss: 59.6571
Epoch 5/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 43s 3ms/step - loss: 59.1641
Epoch 6/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 43s 3ms/step - loss: 59.3526
Epoch 7/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 43s 3ms/step - loss: 58.9287
Epoch 8/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 44s 3ms/step - loss: 58.3894
Epoch 9/9
13678/13678 ━━━━━━━━━━━━━━━━━━━━ 36s 3ms/step - loss: 58.2239
```

Out[ ]:

```
<keras.src.callbacks.history.History at 0x179c0cb7aa0>
```

In [ ]:

```
print("Food Delivery Time Prediction")
a = int(input("Age of Delivery Partner: "))
```

```
b = float(input("Ratings of Previous Deliveries: "))
c = int(input("Total Distance: "))

features = np.array([[a, b, c]])
print("Predicted Delivery Time in Minutes = ", model.predict(features))
```

```
Food Delivery Time Prediction
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Predicted Delivery Time in Minutes =  [[36.346397]]
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js