Food Delivery Time *Prediction*

To predict the food delivery time in real-time, we need to calculate the distance between the food preparation point and the point of food consumption. After finding the distance between the restaurant and the delivery locations, we need to find relationships between the time taken by delivery partners to deliver the food in the past for the same distance.

So, for this task, we need a dataset containing data about the time taken by delivery partners to deliver food from the restaurant to the delivery location. I found an ideal dataset with all the features for this task

- Imports necessary libraries:

- pandas as pd: For handling and analyzing structured data.
- numpy as np: Offers numerical functions, though not used in the current snippet.
- plotly.express as px: For creating interactive plots and visualizations.
- Prints the reference to the head method, but does not execute it.
- This line should be print(data.head()) to actually display the first five rows of the DataFrame.

```
import pandas as pd
import numpy as np
import plotly.express as px

data = pd.read_csv("deliverytime.txt")
print(data.head)
```

```
<bound method NDFrame.head of          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
...     ...                ...                  ...                      ...
45588  7C09      JAPRES04DEL01                   30                      4.8
45589  D641      AGRRES16DEL01                   21                      4.6
45590  4F8D     CHENRES08DEL03                   30                      4.9
45591  5EEE    COIMBRES11DEL01                   20                      4.7
45592  5FB2   RANCHIRES09DEL02                   23                      4.9

       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
...                    ...                   ...                         ...
45588            26.902328             75.794257                   26.912328
45589             0.000000              0.000000                    0.070000
45590            13.022394             80.242439                   13.052394
45591            11.001753             76.986241                   11.041753
45592            23.351058             85.325731                   23.431058

       Delivery_location_longitude Type_of_order Type_of_vehicle  \
0                        75.912471        Snack      motorcycle
1                        77.813237        Snack         scooter
2                        77.688400       Drinks      motorcycle
3                        77.026494       Buffet      motorcycle
4                        80.289982        Snack         scooter
...
45588
45589                     0.070000       Buffet      motorcycle
45590                    80.272439       Drinks         scooter
45591                    77.026241        Snack      motorcycle
```

What can I help you build?

```
45592                    85.405731       Snack        scooter

       Time_taken(min)
0                   24
1                   33
2                   26
3                   21
4                   30
...                ...
45588               32
45589               36
45590               16
45591               26
45592               36

[45593 rows x 11 columns]>
```

Double-click (or enter) to edit

Double-click (or enter) to edit

```
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
```

📊 Dataset Overview and Preliminary Analysis This dataset consists of 45,593 delivery records collected from a food delivery platform. It provides detailed information about the delivery process, including delivery personnel, order types, delivery times, and geographical coordinates.

🔢 Basic Information Total Records: 45,593

Total Features (Columns): 11

Memory Usage: Approximately 3.8 MB

🧾 Feature Description Column Name Description ID Unique identifier for each delivery Delivery_person_ID Unique ID of the delivery person Delivery_person_Age Age of the delivery person (integer) Delivery_person_Ratings Customer rating of the delivery person (float) Restaurant_latitude Latitude of the restaurant Restaurant_longitude Longitude of the restaurant Delivery_location_latitude Latitude of the delivery destination Delivery_location_longitude Longitude of the delivery destination Type_of_order Type of food ordered (e.g., snacks, beverages) Type_of_vehicle Type of vehicle used for delivery (e.g., bike, scooter) Time_taken(min) Time taken to deliver the order in minutes (integer)

🧼 Data Quality Check Missing Values: None. All columns have complete data (no null values).

Data Types:

Categorical: ID, Delivery_person_ID, Type_of_order, Type_of_vehicle

Numerical: Delivery_person_Age, Delivery_person_Ratings, Time_taken(min)

Geospatial: Latitude and longitude values for both restaurant and delivery location are of type float64.

📌 Insights and Potential Use Cases Performance Evaluation: By analyzing the Time_taken(min) in relation to age, ratings, and vehicle type, we can assess what factors contribute to efficient deliveries.

Route Optimization: The latitude and longitude fields allow for distance calculation and geospatial route optimization.

Rating Correlation: Exploring correlations between delivery time and delivery person ratings can help improve customer satisfac

Vehicle Efficiency: Comparing different vehicle types against delivery times can highlight the most efficient transport modes.

```
data.isnull().sum()
```

	0
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

Calculating Distance Between Two Latitudes and Longitudes

📍 Geospatial Distance Calculation In this project, calculating the distance between the restaurant and the delivery location is a key step in understanding delivery dynamics. This was accomplished using the Haversine formula, which calculates the great-circle distance between two points on the Earth's surface based on their latitude and longitude.

📌 Methodology The latitude and longitude values for both the restaurant and the delivery destination were extracted from the dataset:

Restaurant_latitude, Restaurant_longitude

Delivery_location_latitude, Delivery_location_longitude

These coordinates were then used to calculate the straight-line (as-the-crow-flies) distance using the Haversine formula:

$a = \sin\ 2 (\Delta \phi\ 2) + \cos\ (\phi\ 1) \cdot \cos\ (\phi\ 2) \cdot \sin\ 2 (\Delta \lambda\ 2)$ a=sin 2 (2 Δϕ)+cos(ϕ 1)⋅cos(ϕ 2)⋅sin 2 (2 Δλ) $c = 2 \cdot \arctan\ 2 (a , 1 - a)$ c=2⋅arctan2(a, 1−a) Distance = $R \cdot c$ Distance=R⋅c Where:

$\phi\ 1 , \phi\ 2$ ϕ 1,ϕ 2: latitudes in radians

$\lambda\ 1 , \lambda\ 2$ λ 1,λ 2: longitudes in radians

R R: Earth's radius (6371 km)

All degree values were converted to radians using NumPy's efficient vectorized operations to ensure performance across the dataset of over 45,000 records.

```
R = 6371

# Convert 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'])
```

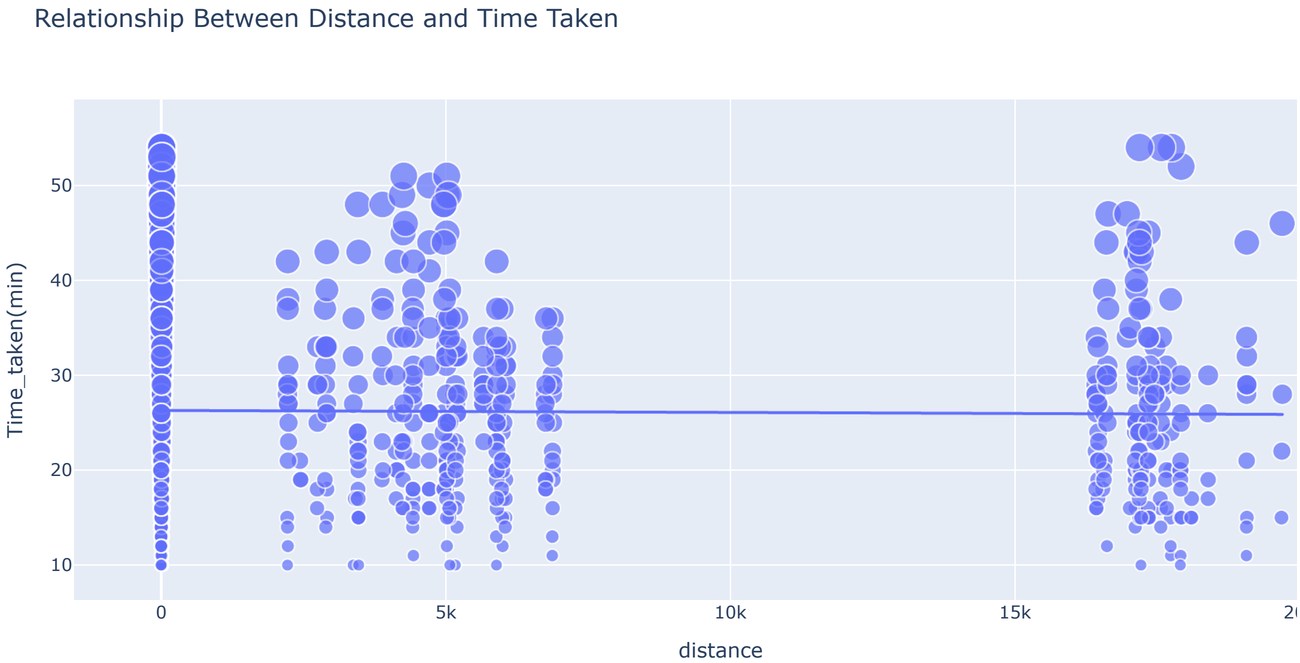
```
print(data.head())
```

```
     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 Exploration

Now let's explore the data to find relationships between the features. I'll start by looking at the relationship between the distance and time taken to deliver the food:

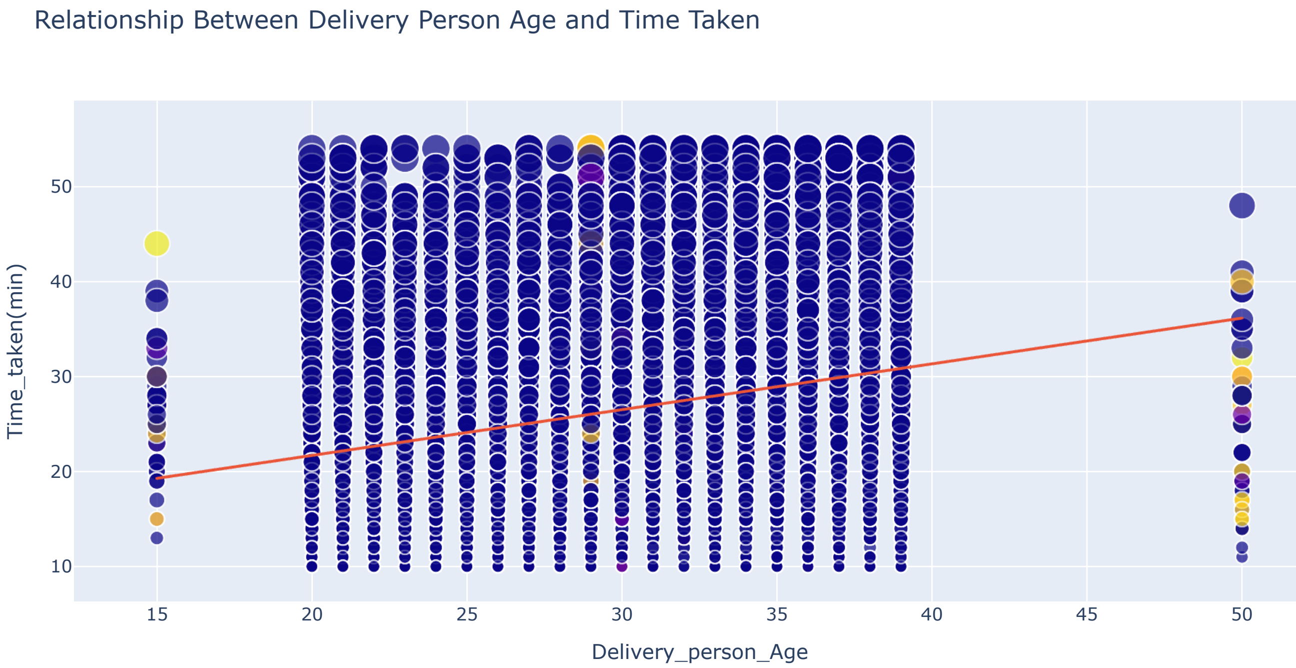
```
figure = px.scatter(data_frame = data,
                    x="distance",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    trendline="ols",
                    title = "Relationship Between Distance and Time Taken")
figure.show()
```



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.

Now let’s have a look 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:

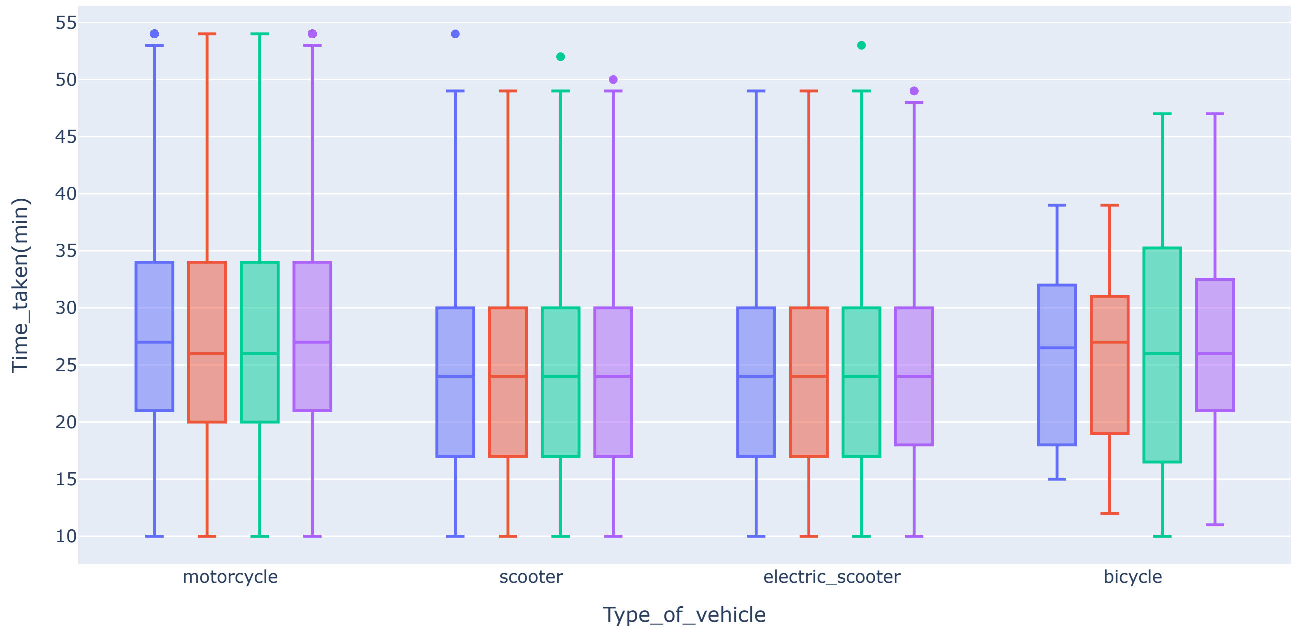
```
figure = px.scatter(data_frame = data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color="distance",
                    trendline="ols",
                    title="Relationship Between Delivery Person Age and Time Taken")
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.

Now let's have a look at the relationship between the time taken to deliver the food and the ratings of the delivery partner:

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

fig.show()
```



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.

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

age of the delivery partner ratings of the delivery partner distance between the restaurant and the delivery location In the section below, I will take you through how to train a Machine Learning model for food delivery time prediction.

Time Prediction Model

Now let's train a Machine Learning model using an LSTM neural network model for the task of food delivery time prediction:

```
#Model
from sklearn.model_selection import train_test_split
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)


#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()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape
```

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)

Detailed Explanation LSTM Layers: LSTMs are capable of learning long-term dependencies. The first LSTM layer processes input sequences and outputs the entire sequence (return_sequences=True), while the second LSTM layer summarizes the sequence to a fixed-size vector.

Dense Layers: These are standard feedforward layers used to map learned representations to the target output. The final dense layer uses a single neuron, making the model ideal for regression or binary classification problems.

Parameter Count:

LSTM layer parameter count includes: weights for input, hidden states, biases (4 × (input_dim + hidden_dim + 1) × hidden_dim).

Dense layers are calculated as: (input_dim + 1) × output_dim.

```
#traning model
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(xtrain,ytrain,batch_size=1,epochs=10)
```

```
Epoch 1/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 206s 5ms/step - loss: 75.4935
Epoch 2/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 261s 5ms/step - loss: 65.5257
Epoch 3/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 264s 5ms/step - loss: 62.0441
Epoch 4/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 261s 5ms/step - loss: 60.9949
Epoch 5/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 265s 5ms/step - loss: 60.5461
Epoch 6/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 260s 5ms/step - loss: 59.1288
Epoch 7/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 261s 5ms/step - loss: 58.8705
Epoch 8/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 263s 5ms/step - loss: 59.1310
Epoch 9/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 261s 5ms/step - loss: 59.6931
Epoch 10/10
41033/41033 ━━━━━━━━━━━━━━━━━━━━ 263s 5ms/step - loss: 58.3871
<keras.src.callbacks.history.History at 0x788b9afe2310>
```

```
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
Age of Delivery Partner: 30
Ratings of Previous Deliveries: 2.8
Total Distance: 6
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 229ms/step
Predicted Delivery Time in Minutes =  [[38.525867]]
```

🧪 **Model Inference and User Interaction**

The final step in the delivery time prediction pipeline involves allowing users (or analysts) to interact with the trained model through a simple input interface. This enables real-time inference by collecting key features and generating delivery time predictions.

🧰 Input Features for Prediction

To make an accurate prediction, the model requires the following three features:

Feature	Description
Age of Delivery Partner	Integer value representing the age of the delivery person
Ratings of Previous Deliveries	Float value representing the average customer rating for the delivery person
Total Distance	Integer or float representing the distance (in kilometers) from the restaurant to the customer

📄 Project Summary: Food Delivery Time Prediction

🎯 **Objective** The primary objective of this project is to build a machine learning model capable of predicting food delivery time based on historical delivery data. This prediction aims to optimize logistics, improve customer satisfaction, and assist delivery platforms in managing operations more efficiently.

📦 **Dataset Overview**

The dataset consists of 45,593 records and 11 features related to delivery operations, including:

Delivery personnel information (age, rating)

Geographic coordinates (restaurant and delivery locations)

Order details (type of order, vehicle used)

Delivery time (in minutes)

The dataset was clean, with no missing values, and structured for direct analysis and modeling.

* 🗺️ *Feature Engineering* *

To enhance the model's predictive power, several derived features were created:

Haversine Distance: Calculated using latitude and longitude to determine the straight-line distance between the restaurant and the delivery location.

Average Speed (optional): Derived from distance and delivery time, useful for performance analysis.

🧠 ** Model Architecture**

A Sequential Neural Network using LSTM (Long Short-Term Memory) layers was employed to capture temporal patterns and dependencies:

LSTM Layers: Two layers to handle input sequences and extract meaningful time-based patterns.

Dense Layers: To map extracted features to the target delivery time.