In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense, LSTM
```

In [2]:
```
Data = pd.read_excel("C:/Users/himan/OneDrive/Desktop/ACADEMIC/SEM - 2/MACHINE LEARNING/Datasheet Online Food
print(Data)
```

	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	75.804257	Meal	motorcycle	
45589	0.070000	Buffet	motorcycle	
45590	80.272439	Drinks	scooter	
45591	77.026241	Snack	motorcycle	
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]
```

In [3]:
```
print(Data.info())
```

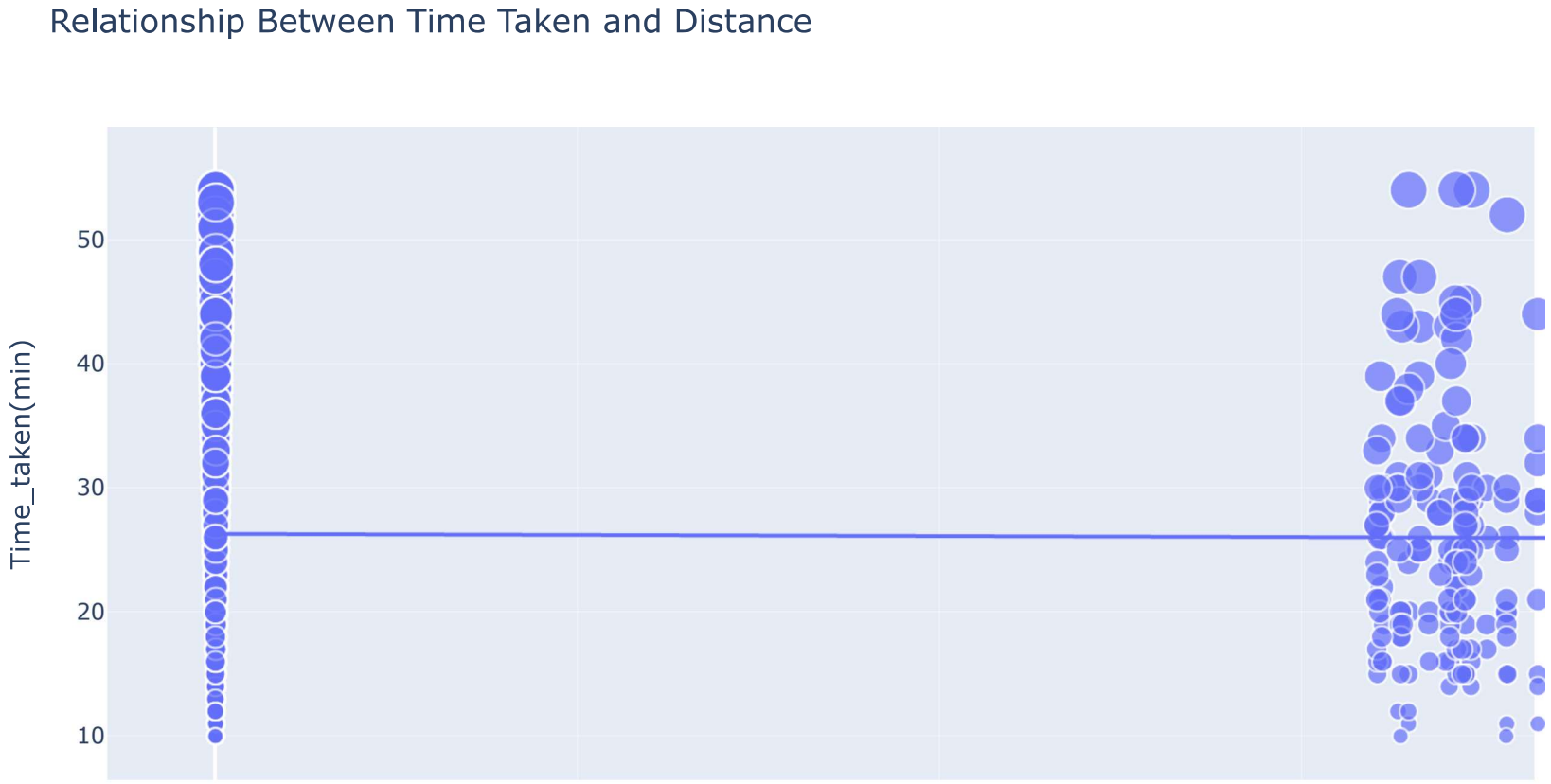
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45593 entries, 0 to 45592
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   ID                           45593 non-null  object 
 1   Delivery_person_ID           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
None
```

In [4]:
```
Data.isnull().sum()
```

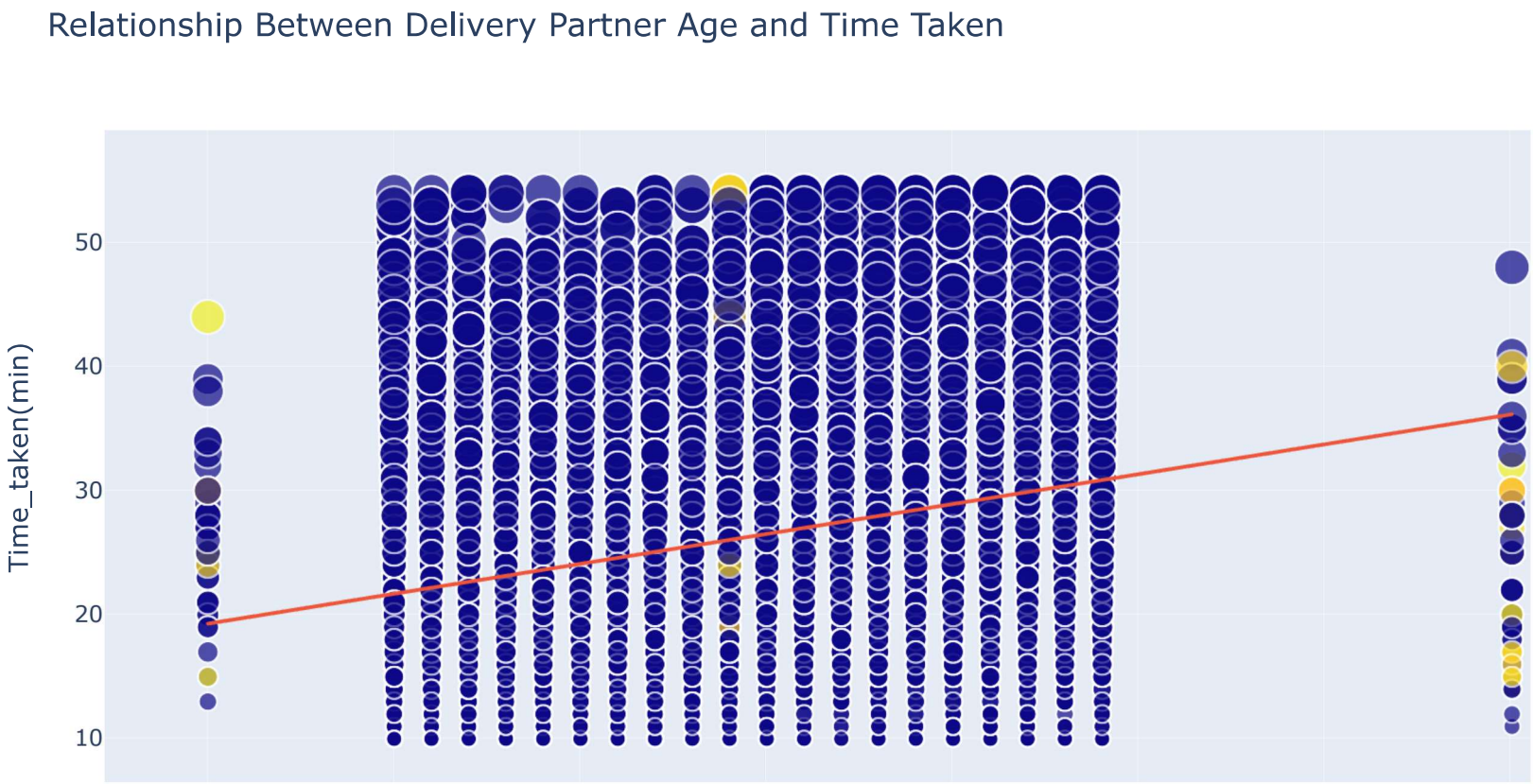
Out[4]:
```
ID                             0
Delivery_person_ID             0
Delivery_person_Age            0
Delivery_person_Ratings        0
Restaurant_latitude            0
Restaurant_longitude           0
Delivery_location_latitude     0
Delivery_location_longitude    0
Type_of_order           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

In [5]:
```
R = 6371  ##The earth's radius (in km)

def deg_to_rad(degrees):
    return degrees * (np.pi/180)

## The haversine formula
def distcalculate(lat1, lon1, lat2, lon2):
    d_lat = deg_to_rad(lat2-lat1)
    d_lon = deg_to_rad(lon2-lon1)
    a1 = np.sin(d_lat/2)**2 + np.cos(deg_to_rad(lat1))
    a2 = np.cos(deg_to_rad(lat2)) * np.sin(d_lon/2)**2
    a = a1 * a2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    return R * c

# Create distance column & calculate the distance
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'])
```

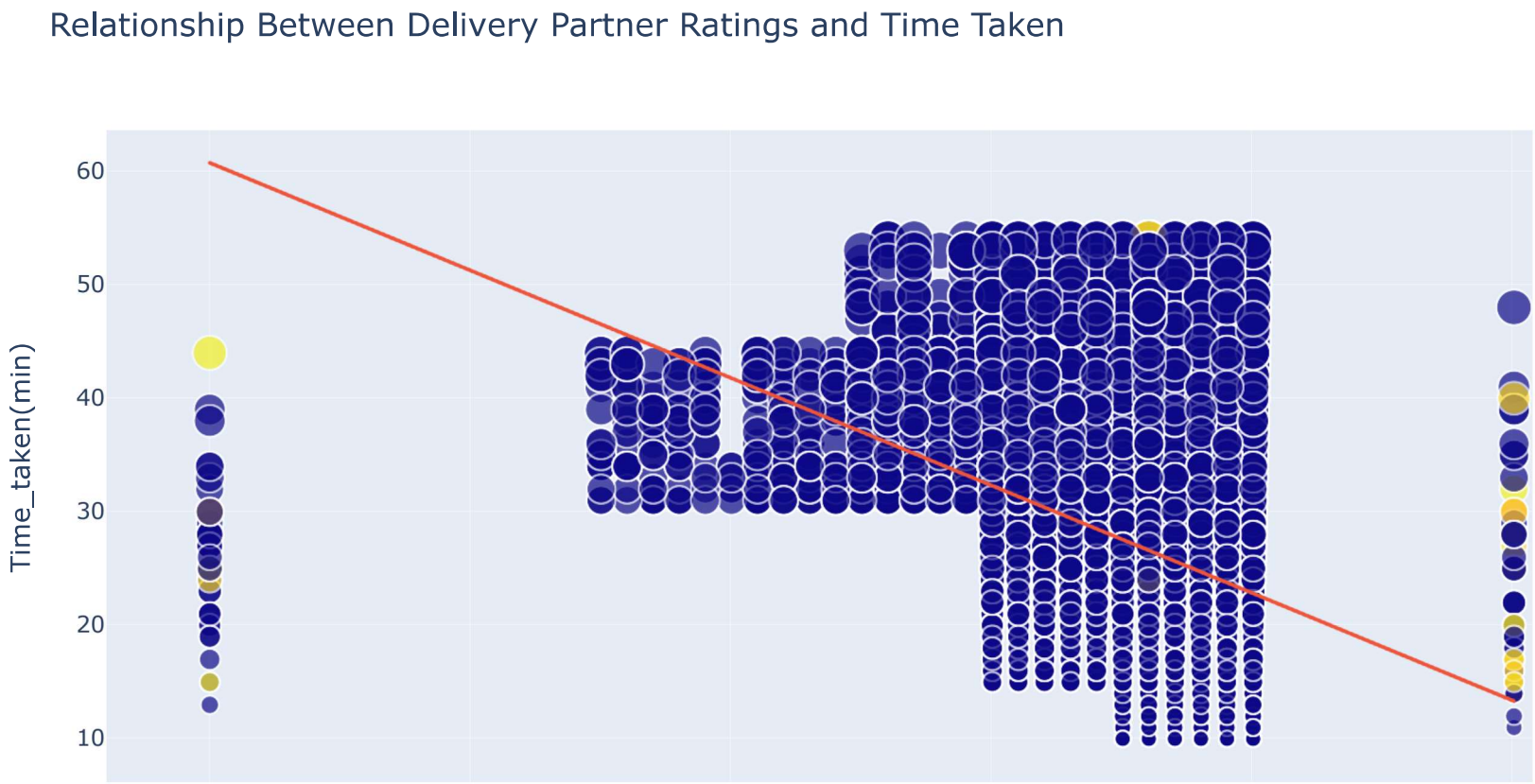
In [6]:
```
figure = px.scatter(data_frame = Data,
                    x="distance",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    trendline="ols",
                    title = "Relationship Between Time Taken and Distance")
figure.show()
```



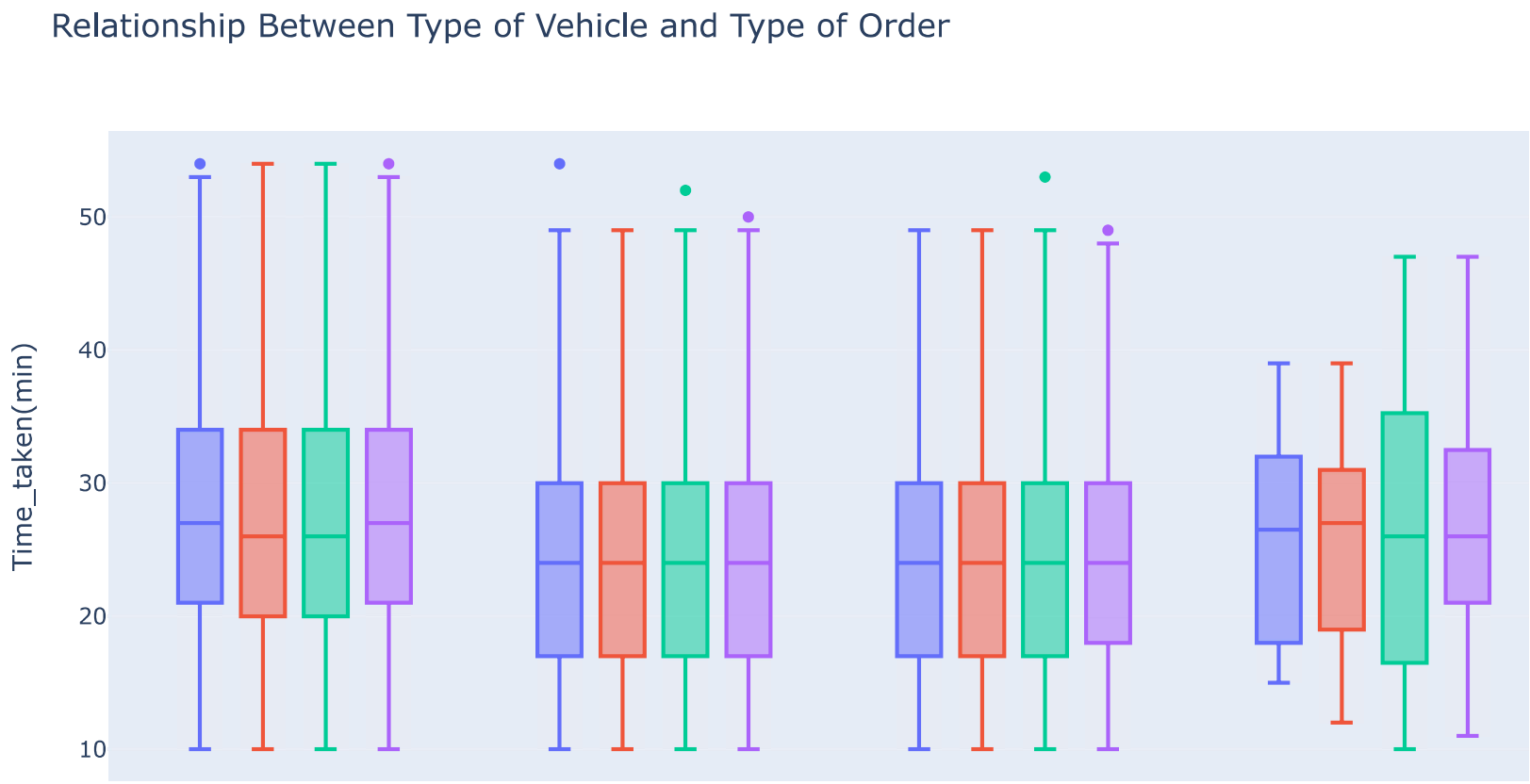
In [7]:
```
figure = px.scatter(data_frame = Data,
                    x="Delivery_person_Age",
                    y="Time_taken(min)",
                    size="Time_taken(min)",
                    color = "distance",
                    trendline="ols",
                    title = "Relationship Between Delivery Partner Age and Time Taken")
figure.show()
```



In [8]:
```
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 Delivery Partner Ratings and Time Taken")
figure.show()
```



In [9]:
```
fig = px.box(Data,
             x="Type_of_vehicle",
             y="Time_taken(min)",
             color="Type_of_order",
             title = "Relationship Between Type of Vehicle and Type of Order")
fig.show()
```



In [10]:
```
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=0.20,
                                                random_state=33)
```

In [11]:
```
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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 3, 128)            66560

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 25)                1625

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 117,619
Trainable params: 117,619
Non-trainable params: 0
_________________________________________________________________
```

In [12]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(xtrain, ytrain, batch_size=1, epochs=9)
```

```
Epoch 1/9
36474/36474 [==============================] - 116s 3ms/step - loss: 69.6281
Epoch 2/9
36474/36474 [==============================] - 112s 3ms/step - loss: 64.7264
Epoch 3/9
36474/36474 [==============================] - 115s 3ms/step - loss: 62.5870
Epoch 4/9
36474/36474 [==============================] - 108s 3ms/step - loss: 61.0355
Epoch 5/9
36474/36474 [==============================] - 101s 3ms/step - loss: 60.2132
Epoch 6/9
36474/36474 [==============================] - 101s 3ms/step - loss: 59.9847
Epoch 7/9
36474/36474 [==============================] - 121s 3ms/step - loss: 59.6463
Epoch 8/9
36474/36474 [==============================] - 113s 3ms/step - loss: 59.1384
Epoch 9/9
36474/36474 [==============================] - 124s 3ms/step - loss: 58.8840
```

Out[12]:
```
<keras.callbacks.History at 0x27d925abc10>
```

In [*]:
```
print("Food Delivery Time Prediction using LSTM")
a = int(input("Delivery Partner Age: "))
b = float(input("Previous Delivery Ratings: "))
c = int(input("Total Distance: "))

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

In [ ]:
```

```