Mini Project Report

Food Delivery Time Prediction Report Submitted to

Jawaharlal Nehru Technological University Anantapur, Ananthapuramu

in partial fulfillment of the requirements for the award
of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

K Vishnu Vardhan	**21121A1250**
K Bharath Kumar	**21121A1251**
K V Siva Reddy	**21121A1252**
K Suchith Reddy	**21121A1253**
Lakhmikanth P	**21121A1254**
M N Nanditha	**21121A1255**
M Sai Jyoshna	**21121A1256**

Department of Information Technology

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(AUTONOMOUS)

(Affiliated to JNTUA, Ananthapuramu, Approved by AICTE, Accredited by NBA & NAAC)

Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

2022-2023

TABLE OF CONTENTS

Institute Vision and Mission

VISION

To be one of the Nation's premier Engineering Colleges by achieving the highest order of excellence in Teaching and Research.

MISSION

- To foster intellectual curiosity, pursuit and dissemination of knowledge.
- To explore students' potential through academic freedom and integrity.
- To promote technical mastery and nurture skilled professionals to face competition in ever increasing complex world.

DEPARTMENT OF INFORMATION TECHNOLOGY

VISION

To become a nationally recognized quality education center in the domain of Computer Science and Information Technology through teaching, training, learning, research and consultancy.

MISSION

- The Department offers undergraduate program in Information Technology to produce high quality information technologists and software engineers by disseminating knowledge through contemporary curriculum, competent faculty and adopting effective teaching-learning methodologies.
- Igniting passion among students for research and innovation by exposing them to real time systems and problems
- Developing technical and life skills in diverse community of students with modern training methods to solve problems in Software Industry.
- Inculcating values to practice engineering in adherence to code of ethics in multicultural and multi discipline teams.

PROGRAM EDUCATIONAL OBJECTIVES

After few years of graduation, the graduates of B. Tech. (IT) Program will be:

1. Enrolled or completed higher education in the core or allied areas of Computer Science and Information Technology or management.
2. Successful entrepreneurial or technical career in the core or allied areas of Computer Science and Information Technology.
3. Continued to learn and to adapt to the world of constantly evolving technologies in the core or allied areas of Computer Science and Information Technology.

PROGRAM OUTCOMES

On successful completion of the Program, the graduates of B. Tech. (IT) Program will be able to:

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics,

natural sciences, and engineering sciences.

3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

On successful completion of the program, the graduates of B.Tech. (IT) program will be able to:

PSO1: Design and develop database systems, apply data analytics techniques, and use advanced databases for data storage, processing and retrieval.

PSO2: Apply network security techniques and tools for the development of highly secure systems.

PSO3: Analyze, design and develop efficient algorithms and software applications to deploy in secure environment to support contemporary services using programming languages, tools and technologies.

PSO4: Apply concepts of computer vision and artificial intelligent for the development of efficient intelligent systems and applications.

INTRODUCTION

Food Delivery services like **Zomato** and **Swiggy** need to show the accurate time it will take to deliver your order to keep transparency with their customers. These companies use **Machine Learning** algorithms to predict the food delivery time based on how much time the delivery partners took for the same distance in the past. So, if you want to learn how to use Machine Learning for food delivery time prediction. In this project will take you through food delivery time prediction with Machine Learning using Python.

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.

Food delivery time prediction is a crucial aspect of the modern food service industry, where customers expect not only delicious meals but also timely and reliable delivery. This computational endeavor involves the development of models that use various factors to estimate the time it will take for a food order to reach its destination. These models rely on a diverse set of inputs, including order details, historical delivery data, and external variables like weather conditions and traffic patterns. By leveraging machine learning techniques, such as regression algorithms, these models analyze the relationships between these factors to make accurate predictions. This predictive capability not only enhances customer satisfaction by providing more accurate delivery estimates but also allows businesses to optimize their operations for efficiency. Additionally, food delivery time prediction models have the potential to contribute significantly to the overall competitiveness and success of food delivery services in an increasingly fast-paced and convenience-driven market.

PROBLEM DEFINITION

This project aims to create a Python-based food delivery time prediction model. The model will take into account various factors like order details, historical delivery data, and external influences such as weather conditions. Data preprocessing will involve cleaning, normalization, and feature encoding. Feature selection will play a crucial role in determining relevant variables for predictions. Machine learning models like Linear Regression, Decision Trees, or Neural Networks will be considered for training. The model's performance will be evaluated using metrics like Mean Absolute Error. Hyperparameter tuning may be used for optimization. Once developed, the model will be integrated for practical use, with ongoing monitoring and maintenance protocols established. Ethical considerations regarding data privacy and fairness will also be taken into account..

DATASET

The dataset you are given here is a cleaned version of the original data on Kaggle. 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..

MODULES

1.Pandas: Certainly! One of the commonly used Python modules is pandas. Pandas is a powerful library for data manipulation and analysis. It provides data structures and functions that make working with structured data, such as tabular data, easier. Here's how to use the pandas module:

Importing Pandas:

To start using pandas, you first import it in your Python script:

- **import pandas as pd**

(It's a common convention to import pandas with the alias pd, making it easier to reference in your code.)

Working with DataFrames:

Pandas primarily works with two main data structures: Series and DataFrames.

Series: A one-dimensional labeled array that can hold data of any type.

DataFrames: A two-dimensional tabular data structure with rows and columns.

Example for how to create a DataFrame and manipulate it:

- import pandas as pd

```
data = {'Name': ['Alice', 'Bob', 'Charlie'],'Age': [25, 30, 35]}
df = pd.DataFrame(data)
print(df)
#Example of reading a CSV file:
import pandas as pd
df = pd.read_csv('data.csv')
```

2.Numpy:

NumPy is a popular Python library for numerical and scientific computing. It provides support for arrays, matrices, and a wide range of mathematical functions. To use NumPy in your Python code, you can import it as follows:

- **import numpy as np**

(In this import statement, ‘numpy’ is the module name and ‘np’ is the alias name of numpy.)

Simple example of using numpy to create a NumPy array and perform some basic operations:

- import numpy as np

```
arr = np.array([1, 2, 3, 4, 5])
sum = np.sum(arr)
max_value = np.max(arr)
```

```
min_value = np.min(arr)
print("Array:", arr)
print("Sum:", sum)
print("Max:", max_value)
print("Min:", min_value)
```

3.Plotly.express:

Plotly Express is a high-level data visualization library in Python that is built on top of Plotly. It provides a simple and concise API for creating interactive and visually appealing charts and plots. To use Plotly Express in your Python code, you typically start by importing it with the following statement:

- **import plotly.express as px**

(In this import statement, 'plotly.express'is the module name and 'px' is the alias name of numpy.) Once you've imported plotly.express, you can use it to create various types of plots, such as scatter plots, line charts, bar charts, and more, with minimal code.Example of creating a scatter plot using Plotly Express:

- import plotly.express as px

```
import pandas as pd
data = pd.DataFrame({'x': [1, 2, 3, 4, 5], 'y': [2, 3, 1, 4, 2]})
# Create a scatter plot
fig = px.scatter(data, x='x', y='y', title='Scatter Plot Example')
# Show the plot
fig.show()
```

4.Keras:

In Python, when working with deep learning models using the Keras library, you can import the Sequential model and the Dense layer from the keras.models and keras.layers modules, respectively. Here's how you can do it:

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

These imports are commonly used when building neural networks with Keras. The Sequential model isa linear stack of layers, and the Dense layer represents a fully connected layer in a neural network. Here's a simple example of how to create a basic feedforward neural network using Keraswith these modules:**Example:**

```
from keras.models import Sequential
```

```
from keras.layers import Dense
model = Sequential() [# Create a Sequential model]
model.add(Dense(units=64, activation='relu', input_dim=100))  [# Add layers to the model]
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

ALGORITHMS:

Code:

```
import pandas as pd
import numpy as np
import plotly.express as px
data = pd.read_csv("DT_DATASET.csv")
print(data.head())
```

op:

```
    ID         D_P_ID  D_P_AGE  D_P_RATING  R_LATITUDE  R_LONGITUDE  \
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

   D_L_LATITUDE  D_L_LONGITUDE T_ORDER   T_VEHICLE  TIME_T(MIN)
0     22.765049      75.912471   SNACK  MOTORCYCLE           24
1     13.043041      77.813237   SNACK     SCOOTER          33
2     12.924264      77.688400  DRINKS  MOTERCYCLE           26
3     11.053669      77.026494  BUFFET  MOTORCYCLE           21
4     13.012793      80.289982   SNACK     SCOOTER          30
```

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

op:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             5 non-null      object
 1   D_P_ID         5 non-null      object
 2   D_P_AGE        5 non-null      int64
 3   D_P_RATING     5 non-null      float64
 4   R_LATITUDE     5 non-null      float64
 5   R_LONGITUDE    5 non-null      float64
```

```
 6   D_L_LATITUDE    5 non-null      float64
 7   D_L_LONGITUDE   5 non-null      float64
 8   T_ORDER         5 non-null      object
 9   T_VEHICLE       5 non-null      object
 10  TIME_T(MIN)     5 non-null      int64
dtypes: float64(5), int64(2), object(4)
memory usage: 568.0+ bytes
```

```
ID              0
D_P_ID          0
D_P_AGE         0
D_P_RATING      0
R_LATITUDE      0
R_LONGITUDE     0
D_L_LATITUDE    0
D_L_LONGITUDE   0
T_ORDER         0
T_VEHICLE       0
TIME_T(MIN)     0
dtype: int64
```

```
# Set the earth's radius (in kilometers)
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, 'R_LATITUDE'],
                                data.loc[i,'R_LONGITUDE'],
                                data.loc[i, 'D_L_LATITUDE'],
                                data.loc[i, 'D_L_LONGITUDE'])
print(data.head())
```

output:

```
  ID         D_P_ID  D_P_AGE  D_P_RATING  R_LATITUDE  R_LONGITUDE  \
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

   D_L_LATITUDE  D_L_LONGITUDE T_ORDER   T_VEHICLE  TIME_T(MIN)   distance
0     22.765049     75.912471   SNACK  MOTORCYCLE          24   3.025149
1     13.043041     77.813237   SNACK     SCOOTER         33  20.183530
2     12.924264     77.688400  DRINKS  MOTERCYCLE          26   1.552758
3     11.053669     77.026494  BUFFET  MOTORCYCLE          21   7.790401
4     13.012793     80.289982   SNACK     SCOOTER         30   6.210138
```

DATA EXPLORATION:

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

figure = px.scatter(data_frame = data,
                    x="D_P_AGE",
                    y="TIME_T(MIN)",
                    size="TIME_T(MIN)",
```

```
            color = "distance",
            trendline="ols",
            title = "Relationship Between Time Taken and Age")
figure.show()

figure = px.scatter(data_frame = data,
            x="D_P_RATING",
            y="TIME_T(MIN)",
            size="TIME_T(MIN)",
            color = "distance",
            trendline="ols",
            title = "Relationship Between Time Taken and Ratings")
figure.show()

fig = px.box(data,
        x="T_VEHICLE",
        y="TIME_T(MIN)",
        color="T_ORDER",
        title="Relationship Between Type of vehicle and Time taken(min)")
fig.show()
```

TRAIN THE MODEL USING MEACHINE LEARNING:

```
#splitting data
from sklearn.model_selection import train_test_split
x = np.array(data[["D_P_AGE",
            "D_P_RATING",
            "distance"]])
y = np.array(data[["TIME_T(MIN)"]])
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                test_size=0.10,
                                random_state=42)

# 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()
```

output:

```
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: 117619 (459.45 KB)
Trainable params: 117619 (459.45 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

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

output:

```
Epoch 1/9
4/4 [==============================] - 4s 16ms/step - loss: 625.0911
Epoch 2/9
4/4 [==============================] - 0s 10ms/step - loss: 558.4596
Epoch 3/9
4/4 [==============================] - 0s 11ms/step - loss: 473.9856
Epoch 4/9
4/4 [==============================] - 0s 11ms/step - loss: 371.1942
Epoch 5/9
4/4 [==============================] - 0s 11ms/step - loss: 242.7551
Epoch 6/9
```

```
4/4 [==============================] - 0s 11ms/step - loss: 134.6994
Epoch 7/9
4/4 [==============================] - 0s 10ms/step - loss: 65.0757
Epoch 8/9
4/4 [==============================] - 0s 10ms/step - loss: 19.6410
Epoch 9/9
4/4 [==============================] - 0s 10ms/step - loss: 13.2870
<keras.src.callbacks.History at 0x7fe4d9a428c0>
```

MAIN MODEL OUTCOME:

```
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))
```

output:

```
Food Delivery Time Prediction
Age of Delivery Partner: 23
Ratings of Previous Deliveries: 4.5
Total Distance: 15
1/1 [==============================] - 1s 797ms/step
Predicted Delivery Time in Minutes =  [[25.723387]]
```

RESULTS

Figure 1: Relationship Between Distance and Time Taken(min)

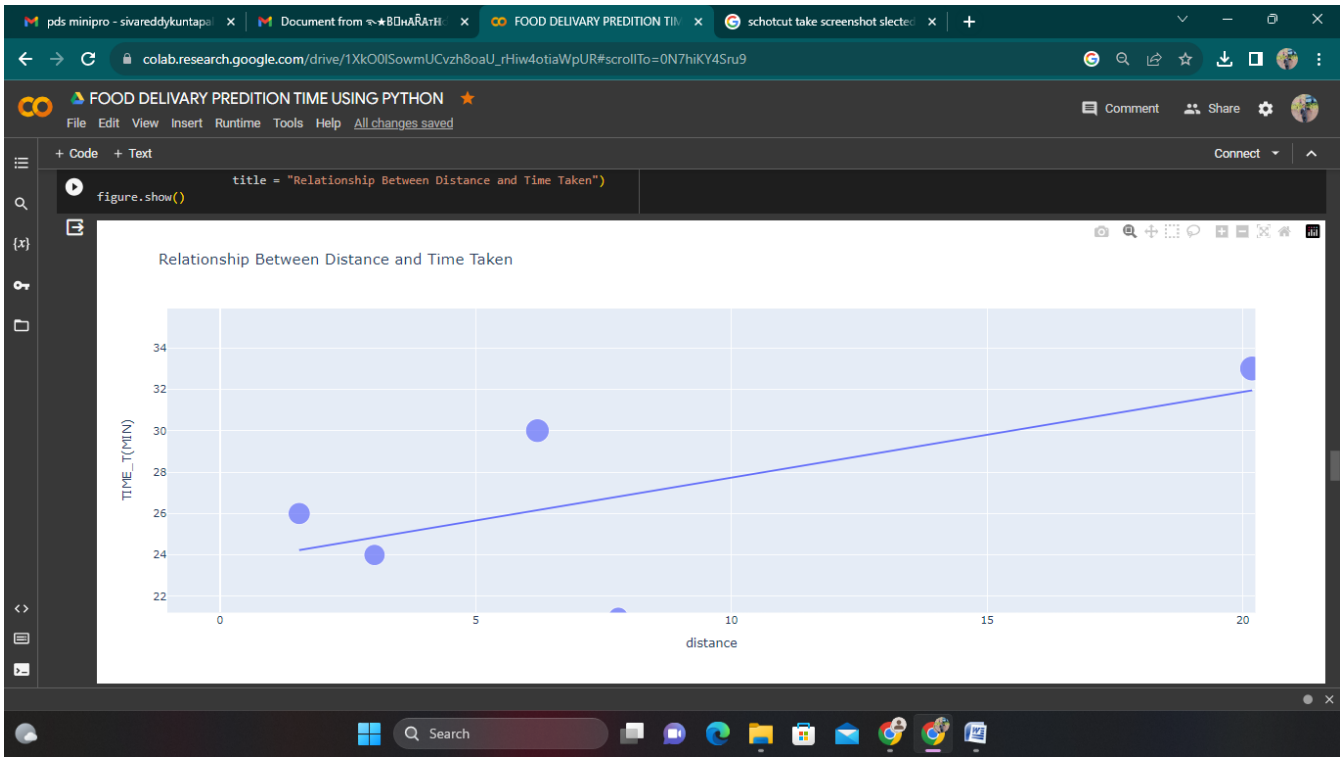


Figure 2 : Relationship Between Time Taken and Age

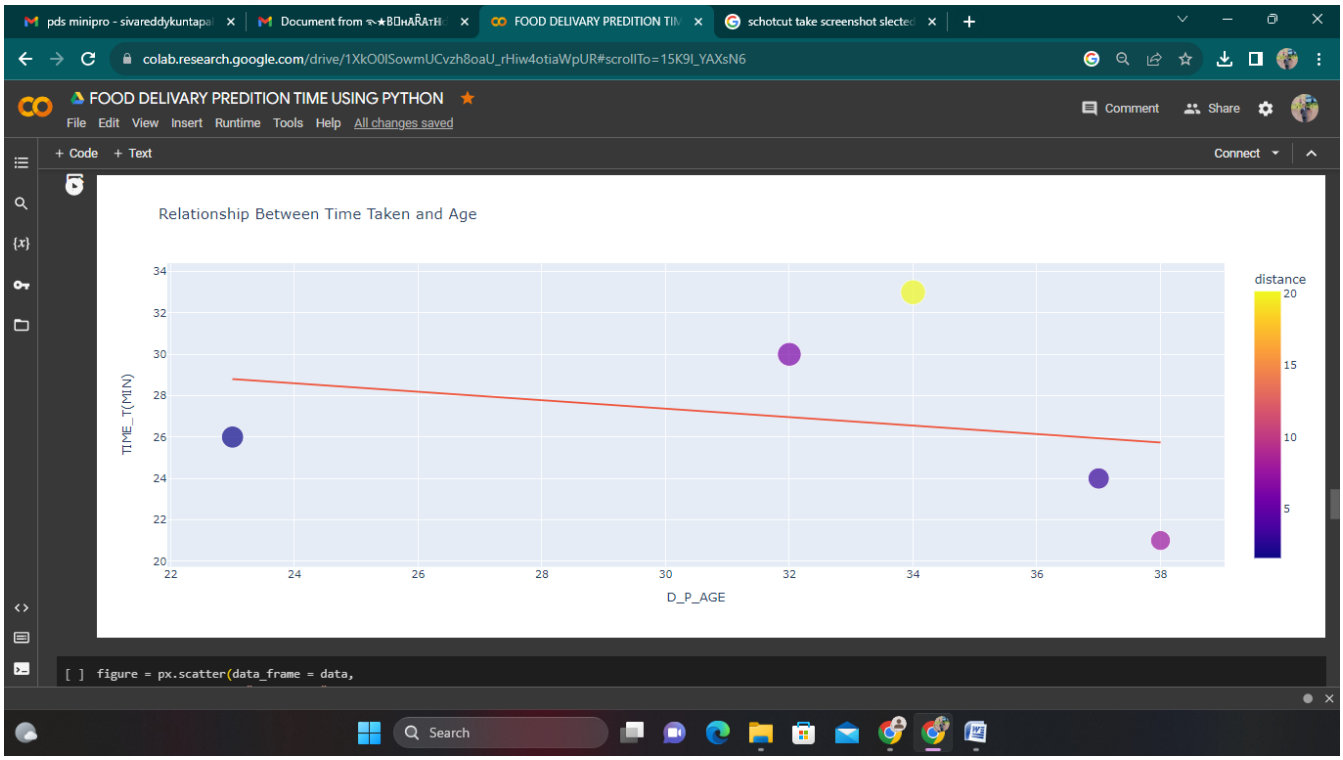


Figure 3 : Relationship Between Time Taken and Ratings

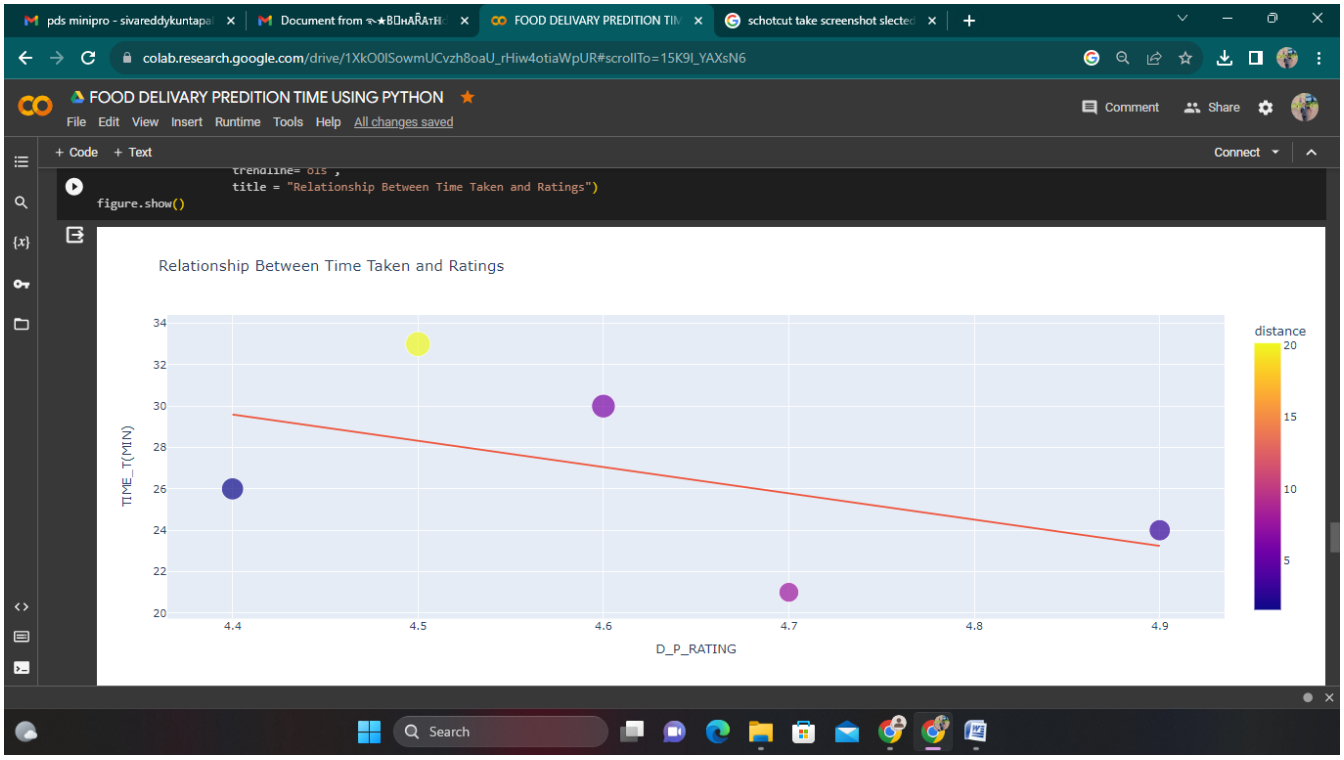
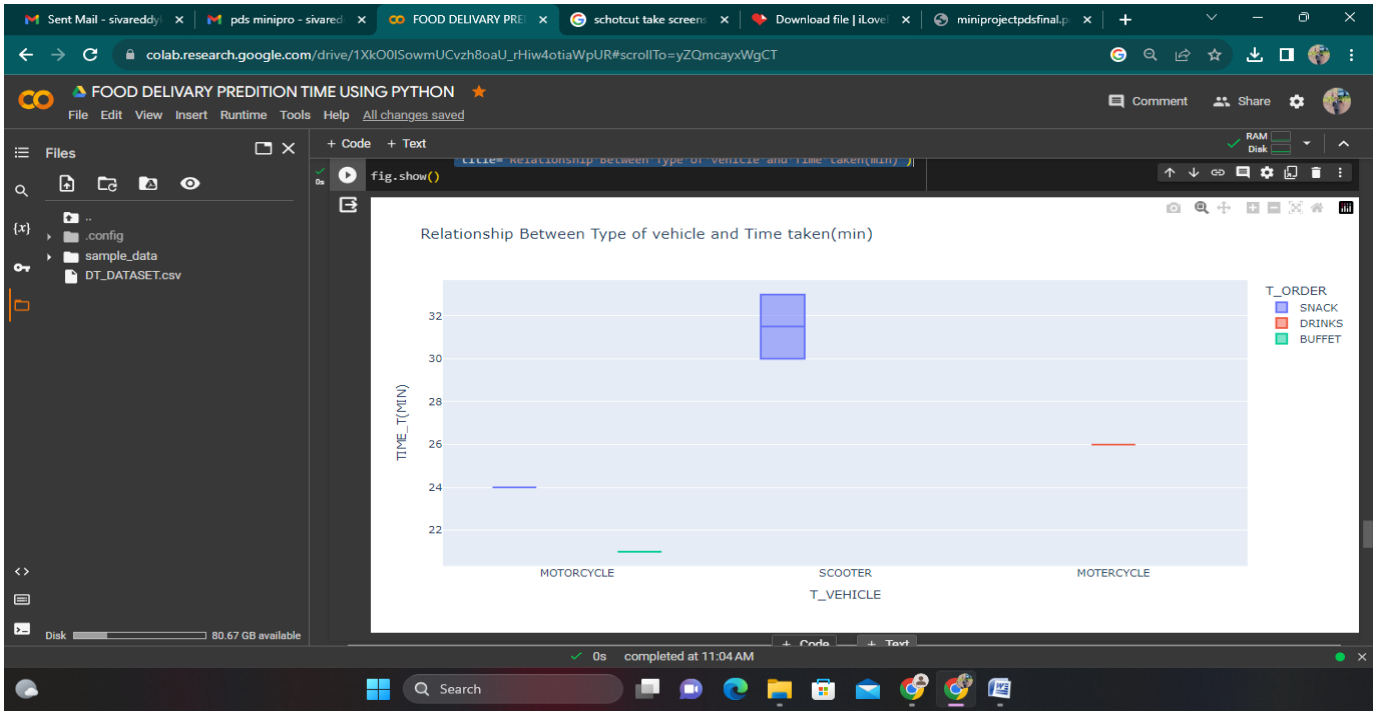


Figure.4 Relationship Between Type of vehicle and Time taken(min)



CONCLUSION

To predict the food delivery time in real time, you 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, you need to find relationships between the time taken by delivery partners to deliver the food in the past for the same distance. I hope you liked this article on food delivery time prediction with Machine Learning using Python.