PREDICTING PALATE PERFECTION: OPTIMIZING FOOD DELIVERY WITH MACHINE LEARNING

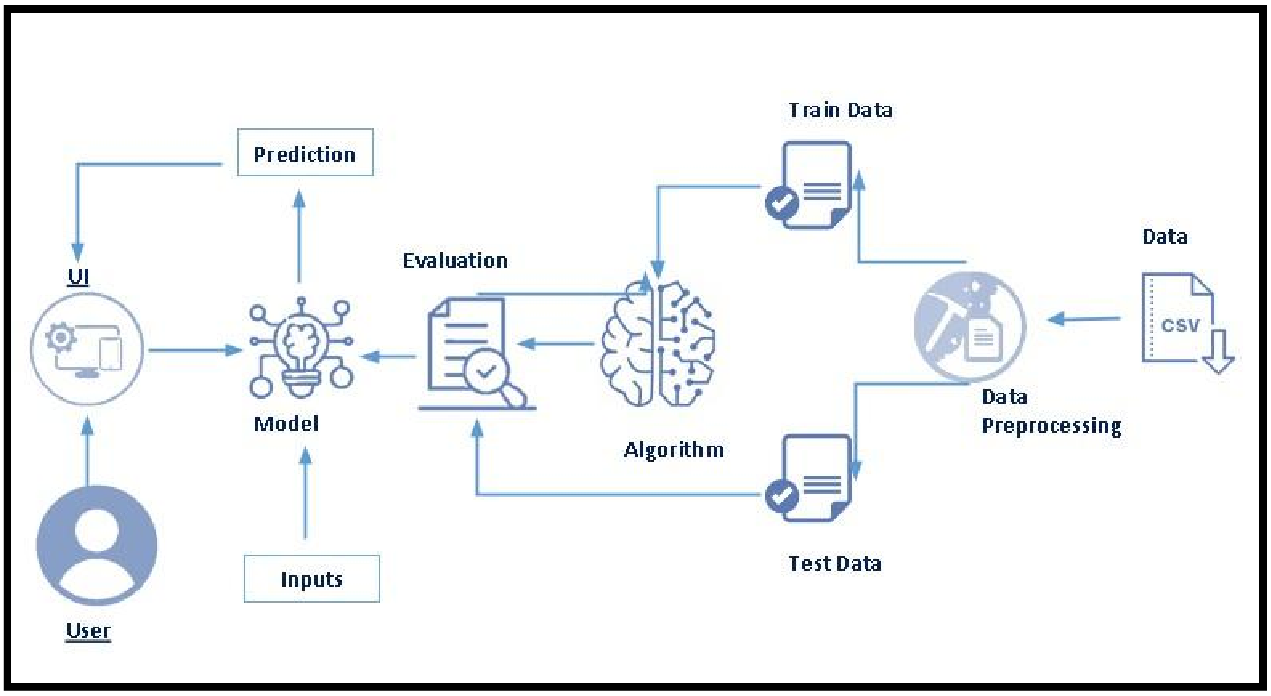
SmartInternz

www.smartinternz.com

Predicting Palate Perfection: Optimizing Food Delivery with Machine Learning

Predicting Palate Perfection: Optimizing Food Delivery with Machine Learning" is a project that aims to improve the efficiency and accuracy of food delivery services using advanced machine learning techniques. By analyzing a wide range of data such as order history, customer preferences, restaurant performance, traffic patterns, and real-time location data, the project seeks to streamline the food delivery process from order placement to delivery. The project focuses on building predictive models to estimate delivery times, optimize delivery routes, and enhance customer satisfaction. These models can help delivery services anticipate delays,manage driver schedules effectively, and pririitize orders based on various factors such as distance, traffic, and order volume.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
    - Specify the business problem
    - Business requirements
    - Literature Survey
    - Social or Business Impact
- Data Collection & Preparation
    - Collect the dataset
    - Data Preparation
- Exploratory Data Analysis
    - Descriptive statistical
    - Visual Analysis
- Model Building
    - Training the model in multiple algorithms
    - Testing the model
- Performance Testing & Hyperparameter Tuning
    - Testing model with multiple evaluation metrics
    - Comapring model accuracy before & after applying hyperparameter tuning
- Model Deployment
    - Save the best model
    - Integrate with Web Framework
- Project Demonstration & Documentation
    - Record explanation Video for project end to end solution
    - Project Documentation-Step by step project development procedure

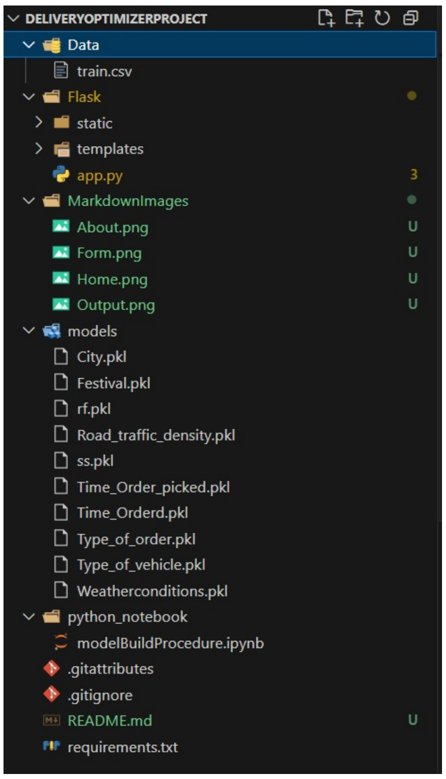
Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- ML Concepts
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to- understand-the-math-behind-xgboost/
- Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



```
DELIVERYOPTIMIZERPROJECT
  Data
    train.csv
  Flask
    static
    templates
    app.py                         3
  MarkdownImages
    About.png                      U
    Form.png                       U
    Home.png                       U
    Output.png                     U
  models
    City.pkl
    Festival.pkl
    rf.pkl
    Road_traffic_density.pkl
    ss.pkl
    Time_Order_picked.pkl
    Time_Orderd.pkl
    Type_of_order.pkl
    Type_of_vehicle.pkl
    Weatherconditions.pkl
  python_notebook
    modelBuildProcedure.ipynb
  .gitattributes
  .gitignore
  README.md                        U
  requirements.txt
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- rf.pkl is our saved model. Further we will use this model for flask integration.
- Train.csv is the Dataset used

- The Notebook file contains procedures for building the model.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Food delivery services face significant challenges in managing unpredictable delivery times, inefficient route planning, and ineffective driver scheduling. These issues lead to delayed orders, increased operational costs, customer dissatisfaction, and negative brand perception. Inaccurate delivery time estimates and poor real-time decision-making further impact service reliability, especially during peak hours and in high-traffic areas. Additionally, limited personalization and order prioritization reduce the overall quality of the customer experience ina highly competitive market.

Activity 2: Business requirements

- The system must be able to predict estimated delivery times for each order by considering factors like restaurant preparation time, distance, traffic conditions, weather, and driver availability.
- The platform must generate optimal delivery routes for drivers in real time to reduce delays, minimize travel distance, and manage traffic constraints.
- The system should prioritze orders based on factors such as order preparation status, customer priority, distance, and order volume to improve service efficiency.
- The system must have the capability to monitor and manage driver schedules, workload distribution and availability in real time to prevent bottlenecks and idle time.

Activity 3: Literature Survey (Student Will Write)

A literature survey for a food delivery optimization project involves researching and reviewing existing studies, articles, and publications on the use of machine learning in delivery services. The survey aims to gather information on current delivery management systems, their strengths, limitations, and the operational challenges they face in real-time route optimization, delivery time prediction, and driver scheduling.The findings from this review will help inform the design, methodology, and implementation strategy for the current project to improve operational efficiency and customer satisfaction in food delivery services.

Activity 4: Social or Business Impact.

Social Impact :- Improved customer experience: By ensuring accurate delivery times and efficient service, this project can enhance customer satisfaction and trust in food delivery platforms.It also supports timely meal access for individuals with health, mobility, or work-related constraints, improving overall community well-being.

Business Model/Impact :- By improving delivery accuracy and route optimization, the project can reduce operational costs and enhance customer satisfaction.This leads to higher customer retention, increased orders, and a stronger competitive position in the food delivery market.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

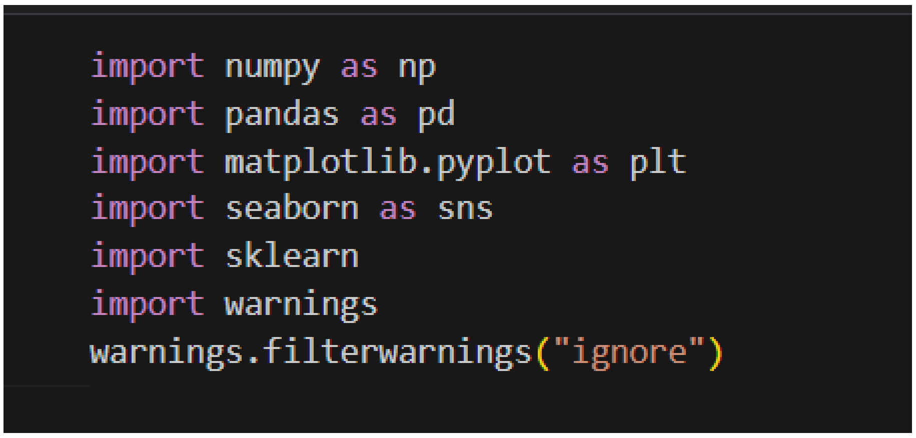
Link: https://www.kaggle.com/datasets/gauravmalik26/food-delivery-dataset?select=train.csv-insurance-claims-data

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.

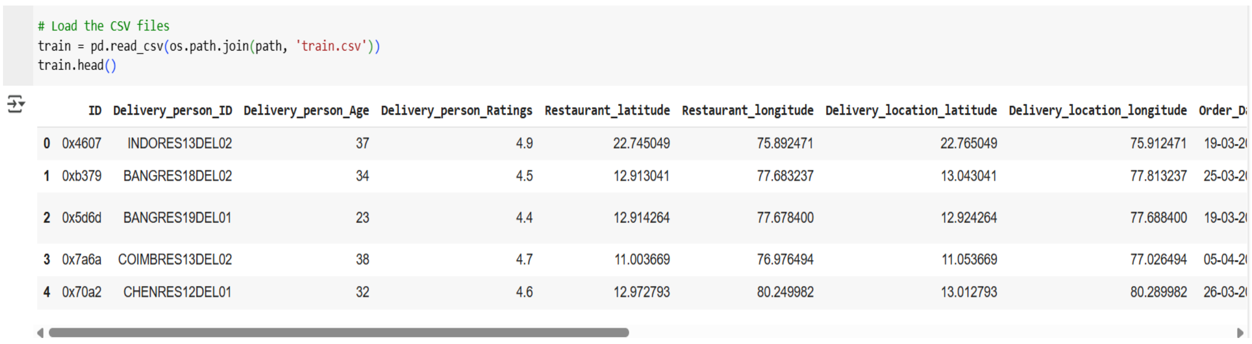


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import warnings
warnings.filterwarnings("ignore")
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.



```
# Load the CSV files
train = pd.read_csv(os.path.join(path, 'train.csv'))
train.head()
```

	ID	Delivery_person_ID	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_location_longitude	Order_D
0	0x4607	INDORES13DEL02	37	4.9	22.745049	75.892471	22.765049	75.912471	19-03-2
1	0xb379	BANGRES18DEL02	34	4.5	12.913041	77.683237	13.043041	77.813237	25-03-2
2	0x5d6d	BANGRES19DEL01	23	4.4	12.914264	77.678400	12.924264	77.688400	19-03-2
3	0x7a6a	COIMBRES13DEL02	38	4.7	11.003669	76.976494	11.053669	77.026494	05-04-2
4	0x70a2	CHENRES12DEL01	32	4.6	12.972793	80.249982	13.012793	80.289982	26-03-2

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

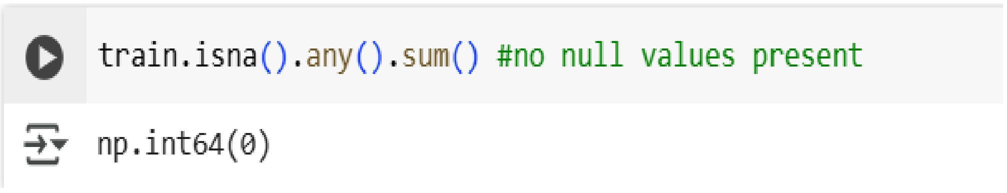
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

- For checking the null values, df.isna().any( ) function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.



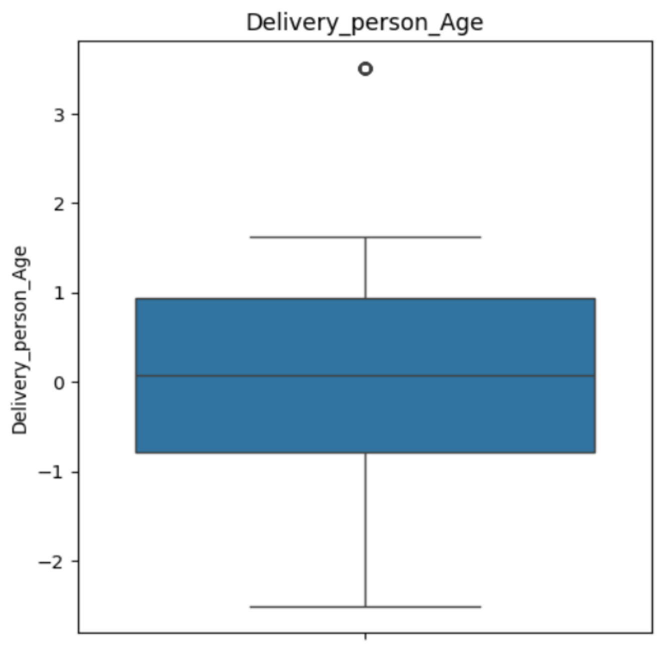
```
train.isna().any().sum() #no null values present
```

```
np.int64(0)
```

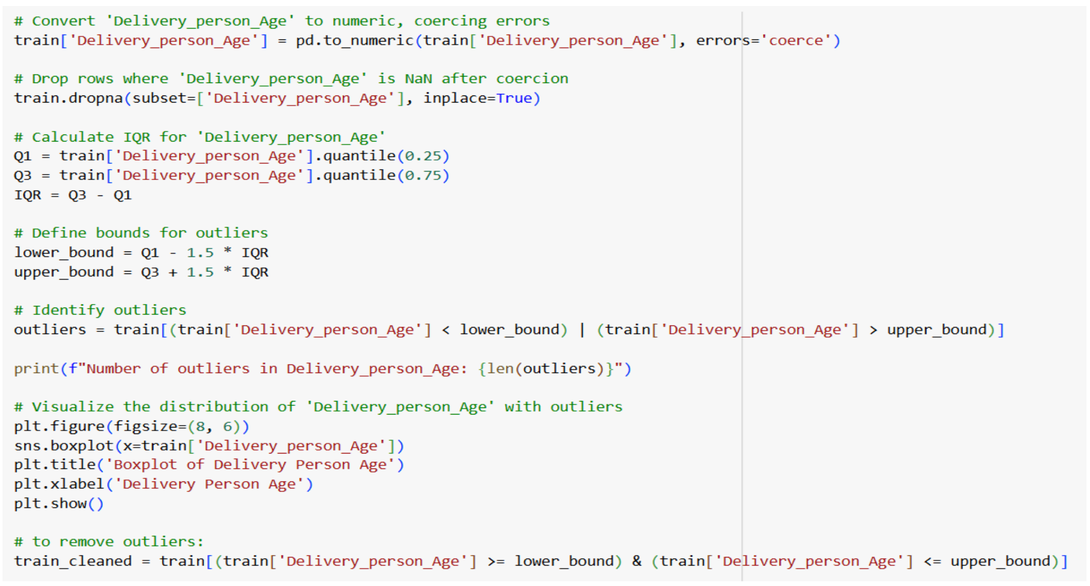
Activity 2.2: Handling Outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of Delivery Person Age feature with some mathematical formula.

- From the below diagram, we could visualize that Delivery Person Age feature has outliers. Boxplot from seaborn library is used here.



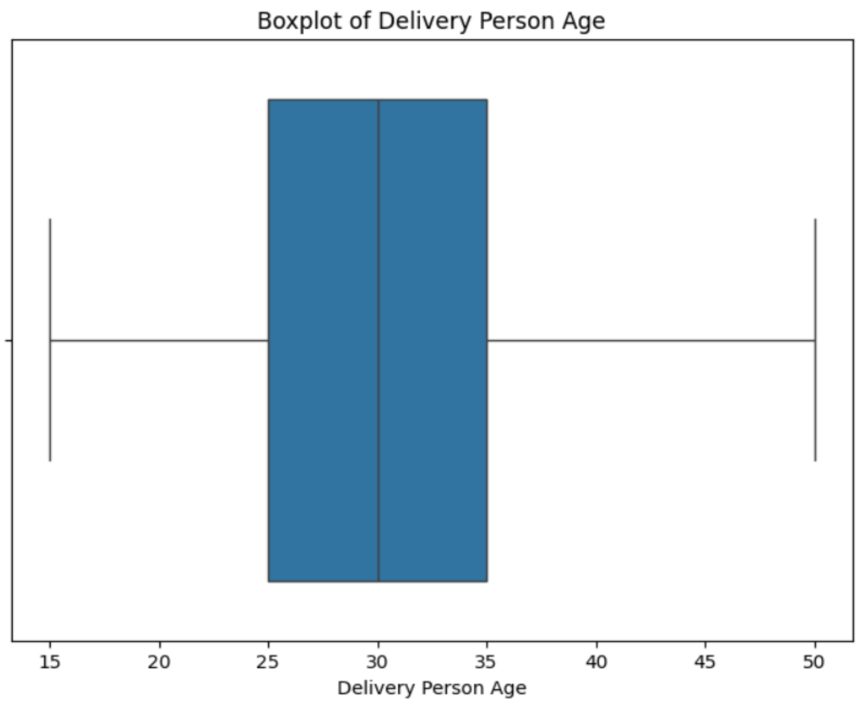
- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference..



```python
# Convert 'Delivery_person_Age' to numeric, coercing errors
train['Delivery_person_Age'] = pd.to_numeric(train['Delivery_person_Age'], errors='coerce')

# Drop rows where 'Delivery_person_Age' is NaN after coercion
train.dropna(subset=['Delivery_person_Age'], inplace=True)

# Calculate IQR for 'Delivery_person_Age'
Q1 = train['Delivery_person_Age'].quantile(0.25)
Q3 = train['Delivery_person_Age'].quantile(0.75)
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = train[(train['Delivery_person_Age'] < lower_bound) | (train['Delivery_person_Age'] > upper_bound)]

print(f"Number of outliers in Delivery_person_Age: {len(outliers)}")

# Visualize the distribution of 'Delivery_person_Age' with outliers
plt.figure(figsize=(8, 6))
sns.boxplot(x=train['Delivery_person_Age'])
plt.title('Boxplot of Delivery Person Age')
plt.xlabel('Delivery Person Age')
plt.show()

# to remove outliers:
train_cleaned = train[(train['Delivery_person_Age'] >= lower_bound) & (train['Delivery_person_Age'] <= upper_bound)]
```

· To handle the outliers transformation technique is used. Here log transformation is used. We have created a function to visualize the distribution and probability plot of Delivery Person Age feature

```
Number of outliers in Delivery_person_Age: 0
```

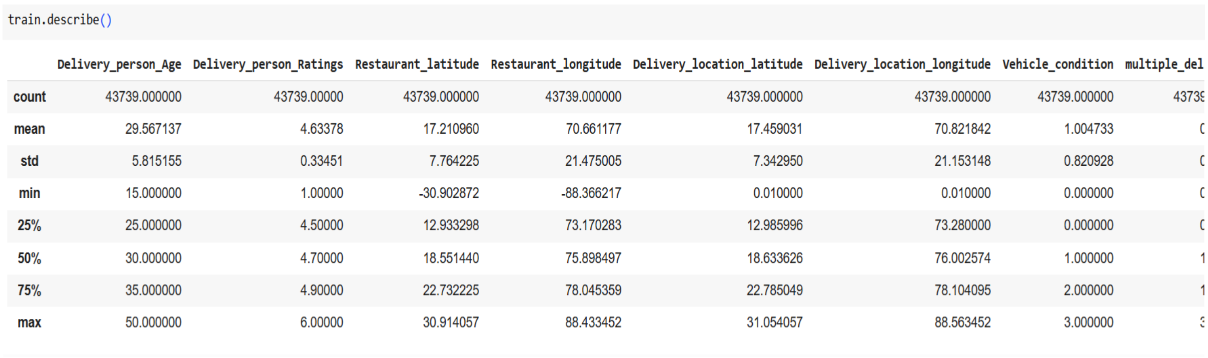


.

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
train.describe()
```

	Delivery_person_Age	Delivery_person_Ratings	Restaurant_latitude	Restaurant_longitude	Delivery_location_latitude	Delivery_location_longitude	Vehicle_condition	multiple_del
count	43739.000000	43739.00000	43739.000000	43739.000000	43739.000000	43739.000000	43739.000000	4373
mean	29.567137	4.63378	17.210960	70.661177	17.459031	70.821842	1.004733	C
std	5.815155	0.33451	7.764225	21.475005	7.342950	21.153148	0.820928	C
min	15.000000	1.00000	-30.902872	-88.366217	0.010000	0.010000	0.000000	C
25%	25.000000	4.50000	12.933298	73.170283	12.985996	73.280000	0.000000	C
50%	30.000000	4.70000	18.551440	75.898497	18.633626	76.002574	1.000000	1
75%	35.000000	4.90000	22.732225	78.045359	22.785049	78.104095	2.000000	1
max	50.000000	6.00000	30.914057	88.433452	31.054057	88.563452	3.000000	3

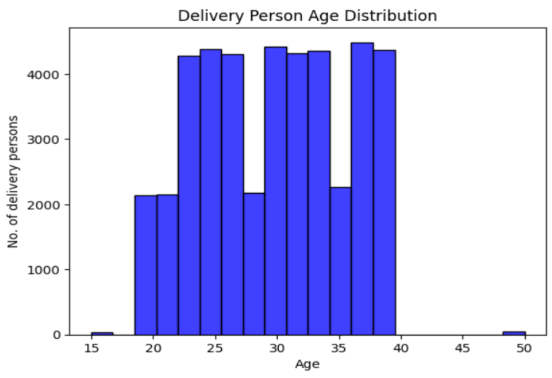
Activity 2: Visual analysis

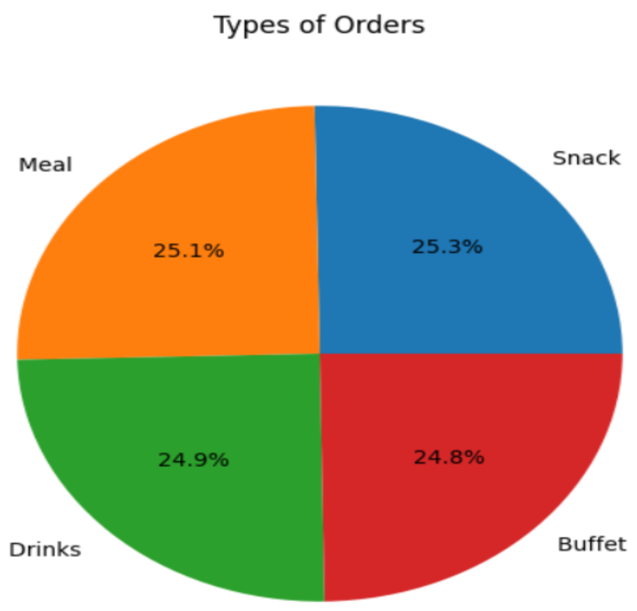
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Piechart and countplot.

Seaborn package provides a wonderful function histogram. With the help of histogram, we can understand the underlying data distributions, identifying patterns. From the histogram, we can say that most of the people are in the age group of 20 to 40.



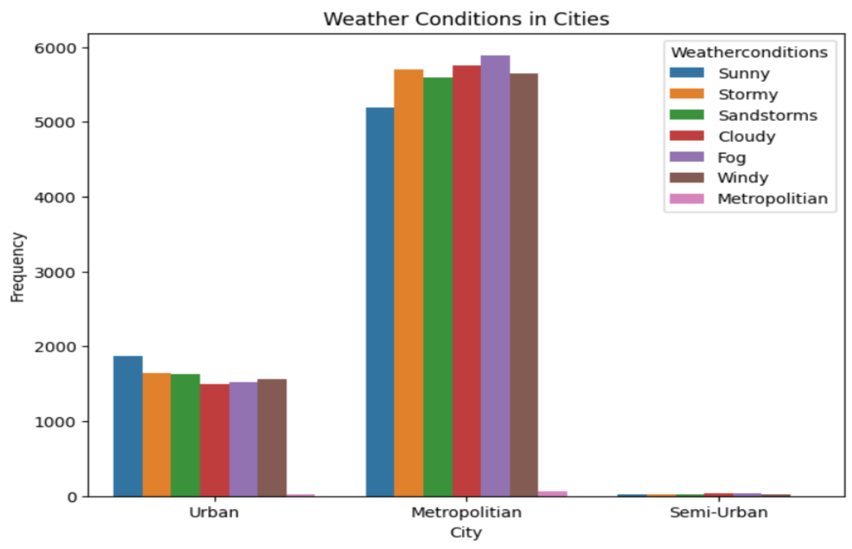


1. Pie chart describes the composition of Types of Orders ordered. It describes the 25.3% of Snacks orders and 25.1% of meal orders and almost equal composition of Buffet and Drinks orders.

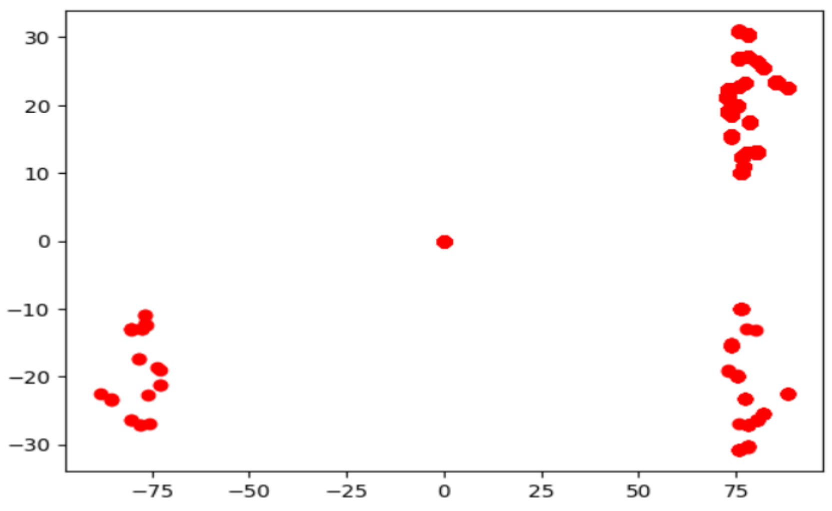
Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we can use:

- A Stacked Barchart is used here.
- From the below plot you can understand the distribution of frequency of different cities in different weather conditions.



· From the below scatterplot we can visualize the realtionship between restaurant's latitude and longitude.



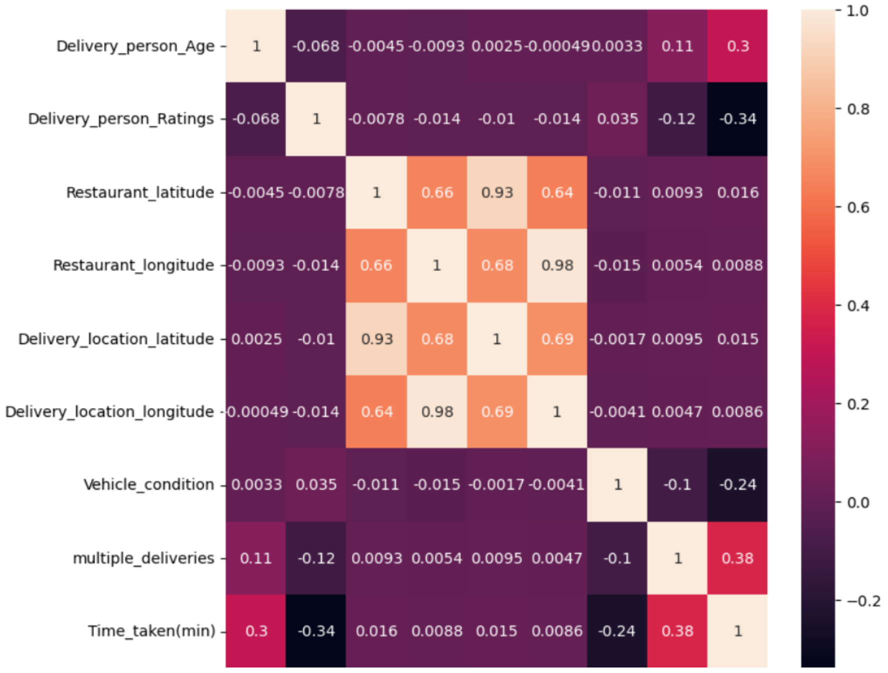
Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.

· From the below image, we came to a conclusion that there are some features which are highly correlated.

· The features like Restaurant_latitude, Restaurant_longitude, delivery_location_latitude and delivery_location_longitude are highly correlated.

· These Highly correlated features should be dropped.



Encoding the Categorical Features:

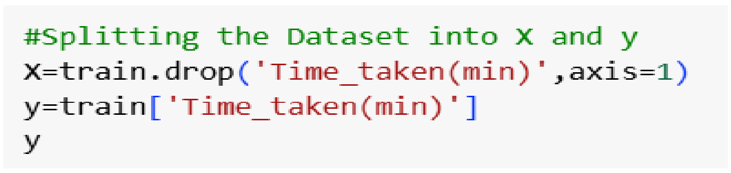
1. The categorical Features are can't be passed directly to the Machine Learning Model. So we convert them into Numerical data based on their order. This Technique is called Encoding.
2. Here we are importing Label Encoder from the Sklearn Library.
3. Here we are applying fit_transform to transform the categorical features to numerical features.

```
import pickle
from sklearn.preprocessing import LabelEncoder

#Creating a dictionary to store label encoders for each categorical column
label_encoders = {}

#Identifying and initializing label encoders for categorical columns
for column in train.columns:
    if train[column].dtype == 'O':#check if the column is of object datatype
        label_encoders[column] = LabelEncoder()
label_encoders
```

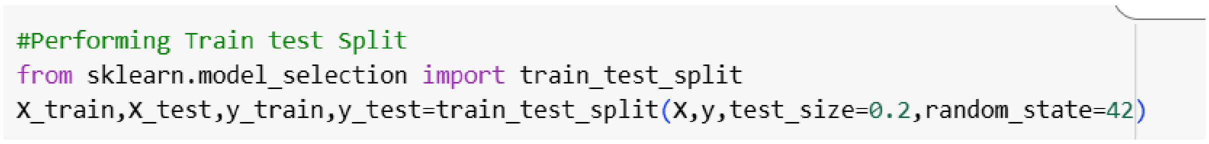
Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



```
#Splitting the Dataset into X and y
X=train.drop('Time_taken(min)',axis=1)
y=train['Time_taken(min)']
y
```

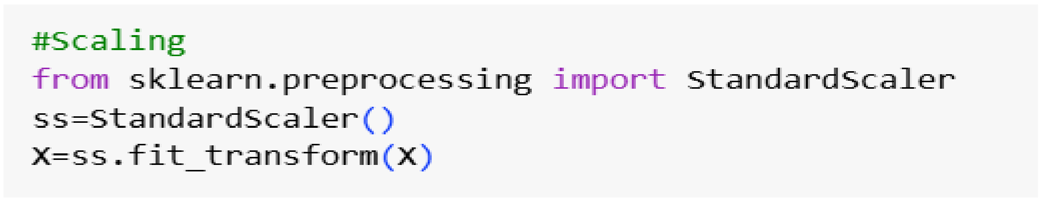


```
#Performing Train test Split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Scaling

1. Scaling is a technique used to transform the values of a dataset to a similar scale to improve the performance of machine learning algorithms. Scaling is important because many machine learning algorithms are sensitive to the scale of the input features.
2. Here we are using Standard Scaler.
3. This scales the data to have a mean of 0 and a standard deviation of 1. The formula is given by:

   X_scaled = (X - X_mean) / X_std



```
#Scaling
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X=ss.fit_transform(X)
```

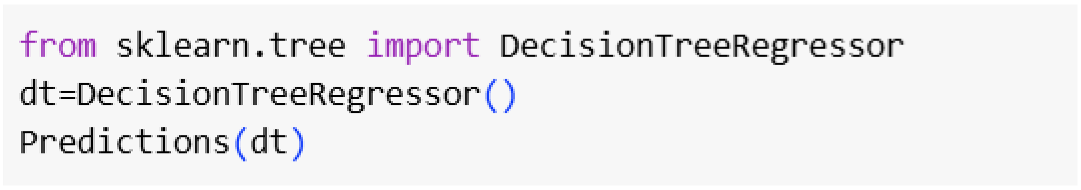
Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

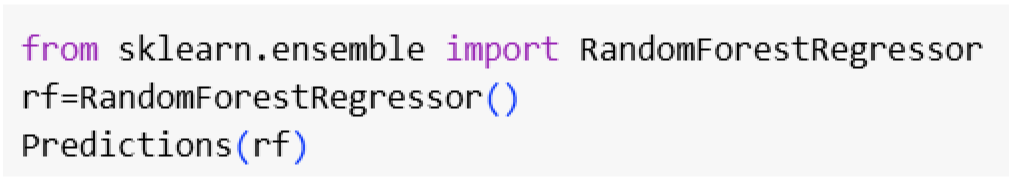
First Decision Tree is imported from sklearn Library then DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.



```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
Predictions(dt)
```

Activity 1.2: Random forest model

First Random Forest Model is imported from sklearn Library then RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. We can find the Train and Test accuracy by X_train and X_test.



```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
Predictions(rf)
```

Activity 1.3: KNN model

KNN Model is imported from sklearn Library then KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
Predictions(knn)
```

Activity 1.4:XG Boost model

Logistic XGBoost is a popular and powerful machine learning algorithm that belongs to the ensemble learning family. It is widely used for both classification and regression tasks and has gained popularity in data science competitions and real-world applications due to its high predictive accuracy and efficiency.

```
from xgboost import XGBRegressor
xg=XGBRegressor()
Predictions(xg)
```

Activity 2: Testing the model

Here we have tested with Random forest algorithm. You can test with all algorithm. With the help of predict() function.

```
models=model.fit(X_train,y_train)
y_pred_test=models.predict(X_test)
y_pred_train=models.predict(X_train)
```

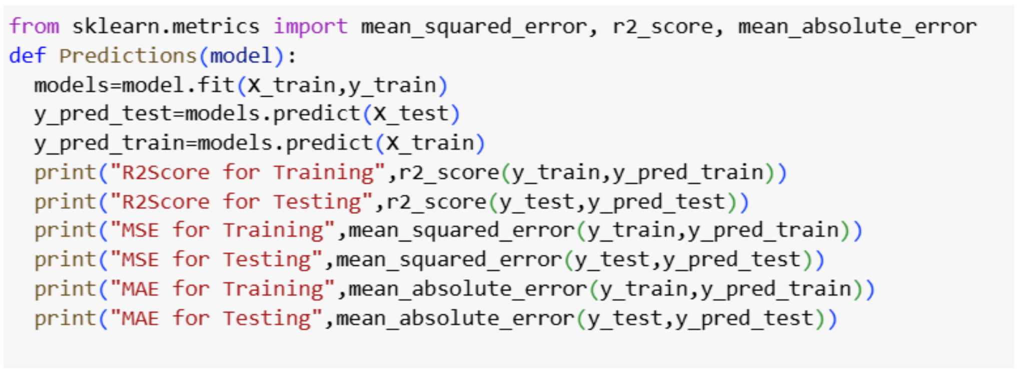
Milestone 5: Performance Testing & Hyperparameter Tuning

Activity 1: Testing model with multiple evaluation metrics

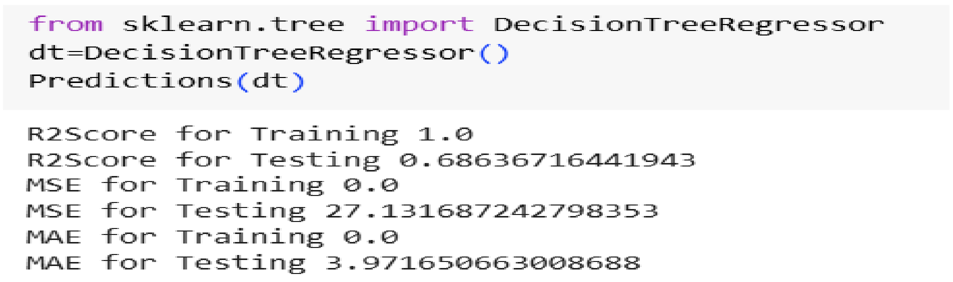
Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Activity 1.1: Compare the model

For comparing the above four models, the predictions function is defined.

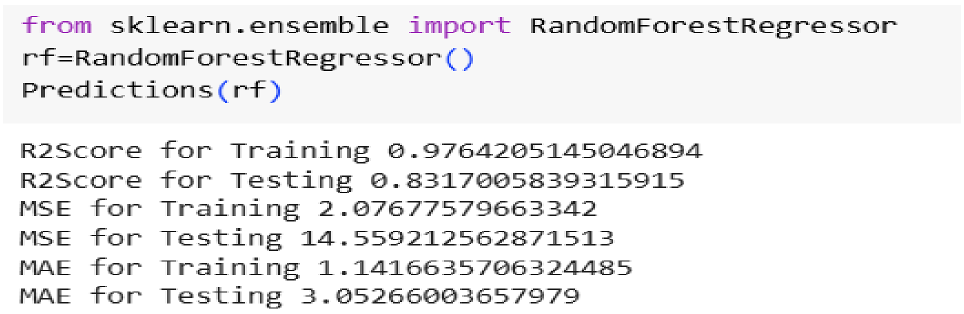


```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
def Predictions(model):
  models=model.fit(X_train,y_train)
  y_pred_test=models.predict(X_test)
  y_pred_train=models.predict(X_train)
  print("R2Score for Training",r2_score(y_train,y_pred_train))
  print("R2Score for Testing",r2_score(y_test,y_pred_test))
  print("MSE for Training",mean_squared_error(y_train,y_pred_train))
  print("MSE for Testing",mean_squared_error(y_test,y_pred_test))
  print("MAE for Training",mean_absolute_error(y_train,y_pred_train))
  print("MAE for Testing",mean_absolute_error(y_test,y_pred_test))
```



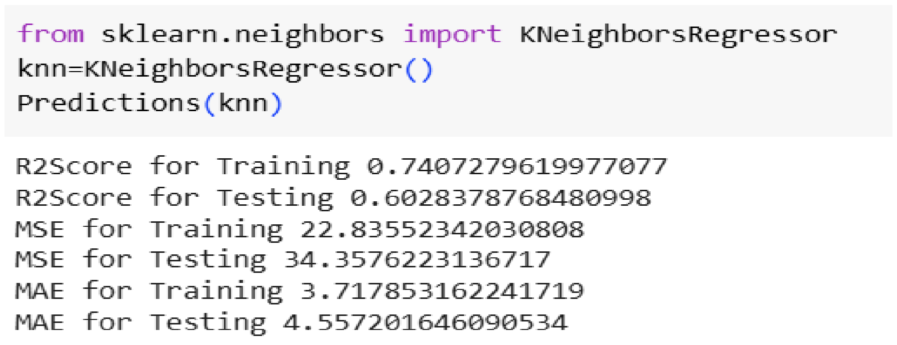
```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
Predictions(dt)
```

```
R2Score for Training 1.0
R2Score for Testing 0.686367164419443
MSE for Training 0.0
MSE for Testing 27.131687242798353
MAE for Training 0.0
MAE for Testing 3.971650663008688
```



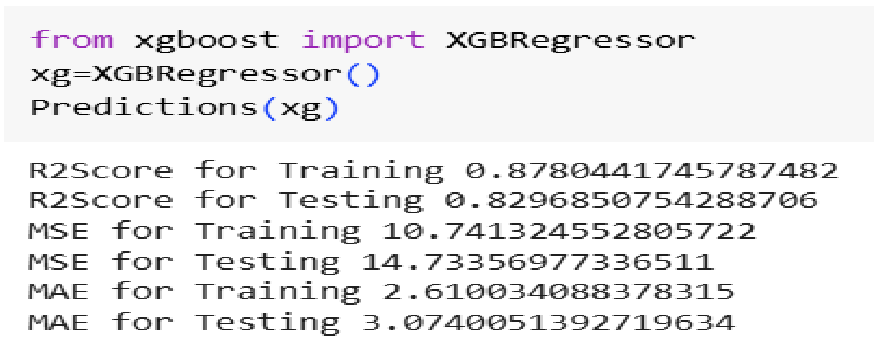
```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
Predictions(rf)
```

```
R2Score for Training 0.9764205145046894
R2Score for Testing 0.8317005839315915
MSE for Training 2.07677579663342
MSE for Testing 14.559212562871513
MAE for Training 1.1416635706324485
MAE for Testing 3.05266003657979
```



```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
Predictions(knn)
```

```
R2Score for Training 0.7407279619977077
R2Score for Testing 0.6028378768480998
MSE for Training 22.83552342030808
MSE for Testing 34.3576223136717
MAE for Training 3.717853162241719
MAE for Testing 4.557201646090534
```



```
from xgboost import XGBRegressor
xg=XGBRegressor()
Predictions(xg)
```

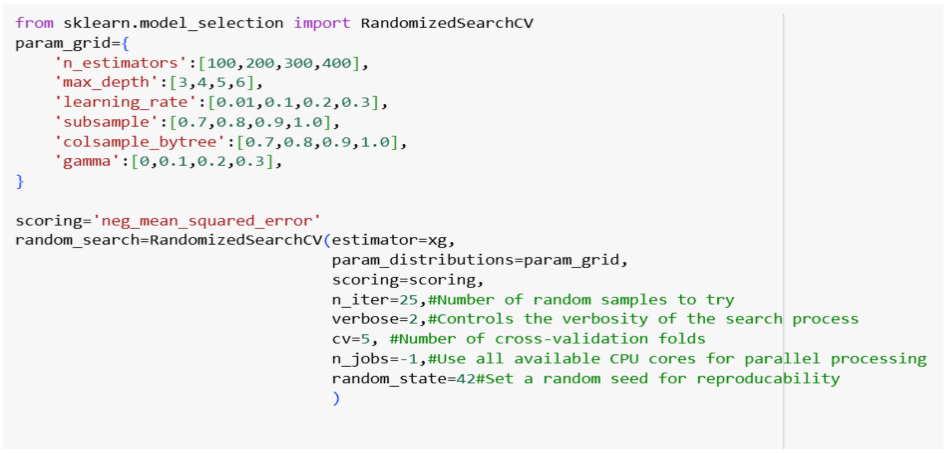
```
R2Score for Training 0.8780441745787482
R2Score for Testing 0.8296850754288706
MSE for Training 10.741324552805722
MSE for Testing 14.73356977336511
MAE for Training 2.610034088378315
MAE for Testing 3.0740051392719634
```

After calling the function, the results of models are displayed as output. From the above models Random Forest is performing well.

Activity 2: Comparing model accuracy before & after applying hyperparameter tuning (Hyperparameter tuning is optional. For this project it is not required.)

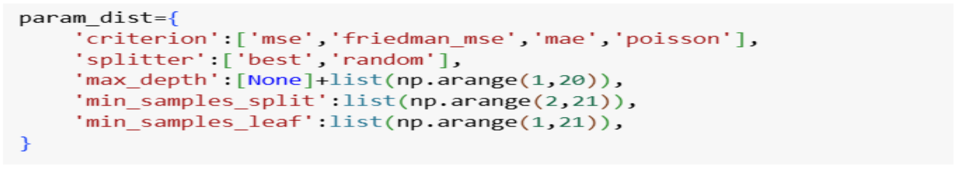
Evaluating performance of the model From sklearn, RandomizedSearchCV is used to evaluate the score of the model.

Activity 2.1: Implementing RandomSearchCV for XGBoost

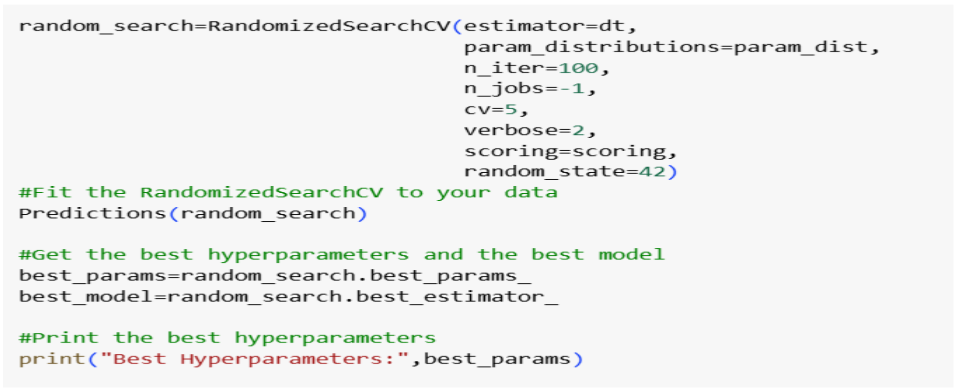


```
from sklearn.model_selection import RandomizedSearchCV
param_grid={
    'n_estimators':[100,200,300,400],
    'max_depth':[3,4,5,6],
    'learning_rate':[0.01,0.1,0.2,0.3],
    'subsample':[0.7,0.8,0.9,1.0],
    'colsample_bytree':[0.7,0.8,0.9,1.0],
    'gamma':[0,0.1,0.2,0.3],
}

scoring='neg_mean_squared_error'
random_search=RandomizedSearchCV(estimator=xg,
                                 param_distributions=param_grid,
                                 scoring=scoring,
                                 n_iter=25,#Number of random samples to try
                                 verbose=2,#Controls the verbosity of the search process
                                 cv=5, #Number of cross-validation folds
                                 n_jobs=-1,#Use all available CPU cores for parallel processing
                                 random_state=42#Set a random seed for reproducability
                                 )
```

Activity 2.2: Implementing RandomSearchCV for DecisionTree

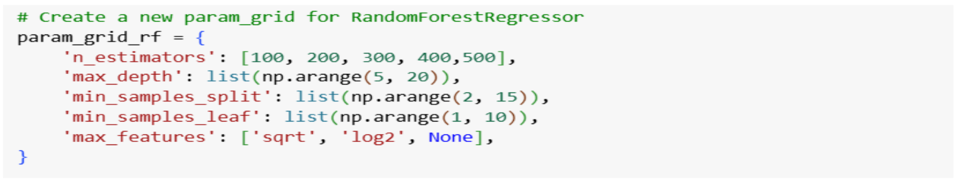


```
param_dist={
     'criterion':['mse','friedman_mse','mae','poisson'],
     'splitter':['best','random'],
     'max_depth':[None]+list(np.arange(1,20)),
     'min_samples_split':list(np.arange(2,21)),
     'min_samples_leaf':list(np.arange(1,21)),
}
```

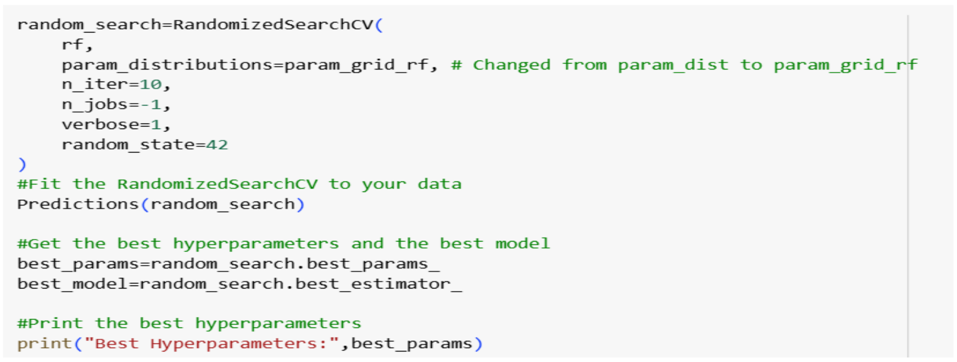


```
random_search=RandomizedSearchCV(estimator=dt,
                                 param_distributions=param_dist,
                                 n_iter=100,
                                 n_jobs=-1,
                                 cv=5,
                                 verbose=2,
                                 scoring=scoring,
                                 random_state=42)
#Fit the RandomizedSearchCV to your data
Predictions(random_search)

#Get the best hyperparameters and the best model
best_params=random_search.best_params_
best_model=random_search.best_estimator_

#Print the best hyperparameters
print("Best Hyperparameters:",best_params)
```

Activity 2.3: Implementing RandomSearchCV for RandomForest



```
# Create a new param_grid for RandomForestRegressor
param_grid_rf = {
    'n_estimators': [100, 200, 300, 400,500],
    'max_depth': list(np.arange(5, 20)),
    'min_samples_split': list(np.arange(2, 15)),
    'min_samples_leaf': list(np.arange(1, 10)),
    'max_features': ['sqrt', 'log2', None],
}
```

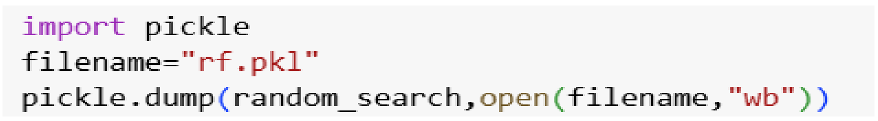


```
random_search=RandomizedSearchCV(
    rf,
    param_distributions=param_grid_rf, # Changed from param_dist to param_grid_rf
    n_iter=10,
    n_jobs=-1,
    verbose=1,
    random_state=42
)
#Fit the RandomizedSearchCV to your data
Predictions(random_search)

#Get the best hyperparameters and the best model
best_params=random_search.best_params_
best_model=random_search.best_estimator_

#Print the best hyperparameters
print("Best Hyperparameters:",best_params)
```

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance.This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
import pickle
filename="rf.pkl"
pickle.dump(random_search,open(filename,"wb"))
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

1. Building HTML Pages
2. Building server-side script
3. Run the web application

Activity 2.1: Building Html Page:

For this project create HTML files namely

- index.html

  about.html

  navbar.html

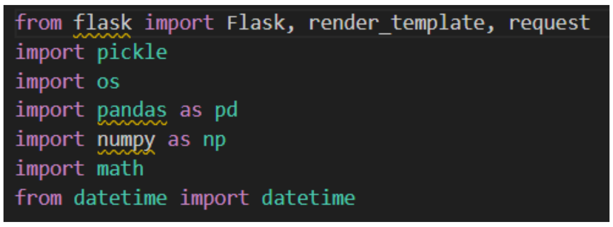
  predict.html

  output.html

and save them in the templates folder. Refer this link for templates.

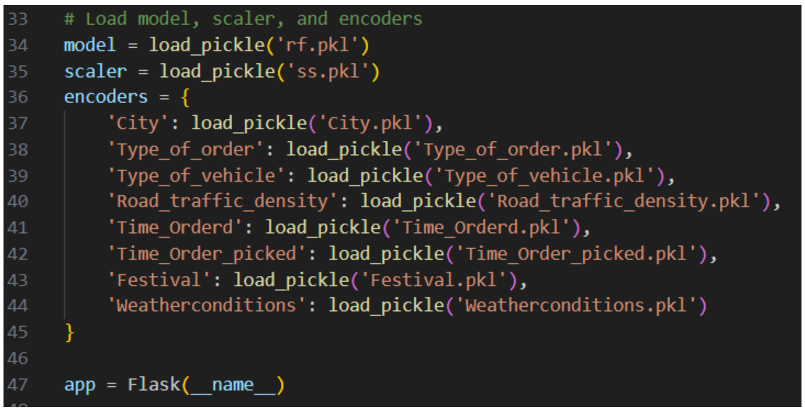
Activity 2.2: Build Python code:

Import the libraries



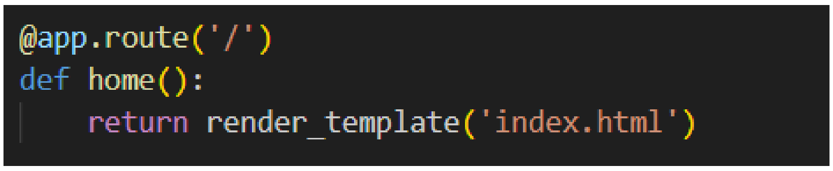
```python
from flask import Flask, render_template, request
import pickle
import os
import pandas as pd
import numpy as np
import math
from datetime import datetime
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.



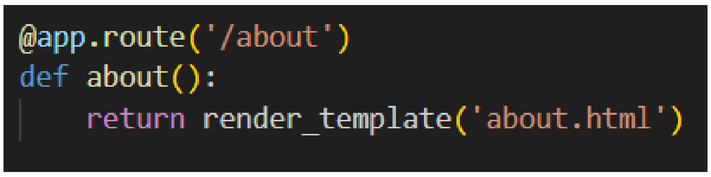
```python
# Load model, scaler, and encoders
model = load_pickle('rf.pkl')
scaler = load_pickle('ss.pkl')
encoders = {
    'City': load_pickle('City.pkl'),
    'Type_of_order': load_pickle('Type_of_order.pkl'),
    'Type_of_vehicle': load_pickle('Type_of_vehicle.pkl'),
    'Road_traffic_density': load_pickle('Road_traffic_density.pkl'),
    'Time_Orderd': load_pickle('Time_Orderd.pkl'),
    'Time_Order_picked': load_pickle('Time_Order_picked.pkl'),
    'Festival': load_pickle('Festival.pkl'),
    'Weatherconditions': load_pickle('Weatherconditions.pkl')
}

app = Flask(__name__)
```

Render HTML page:



```python
@app.route('/')
def home():
    return render_template('index.html')
```

Render About page:



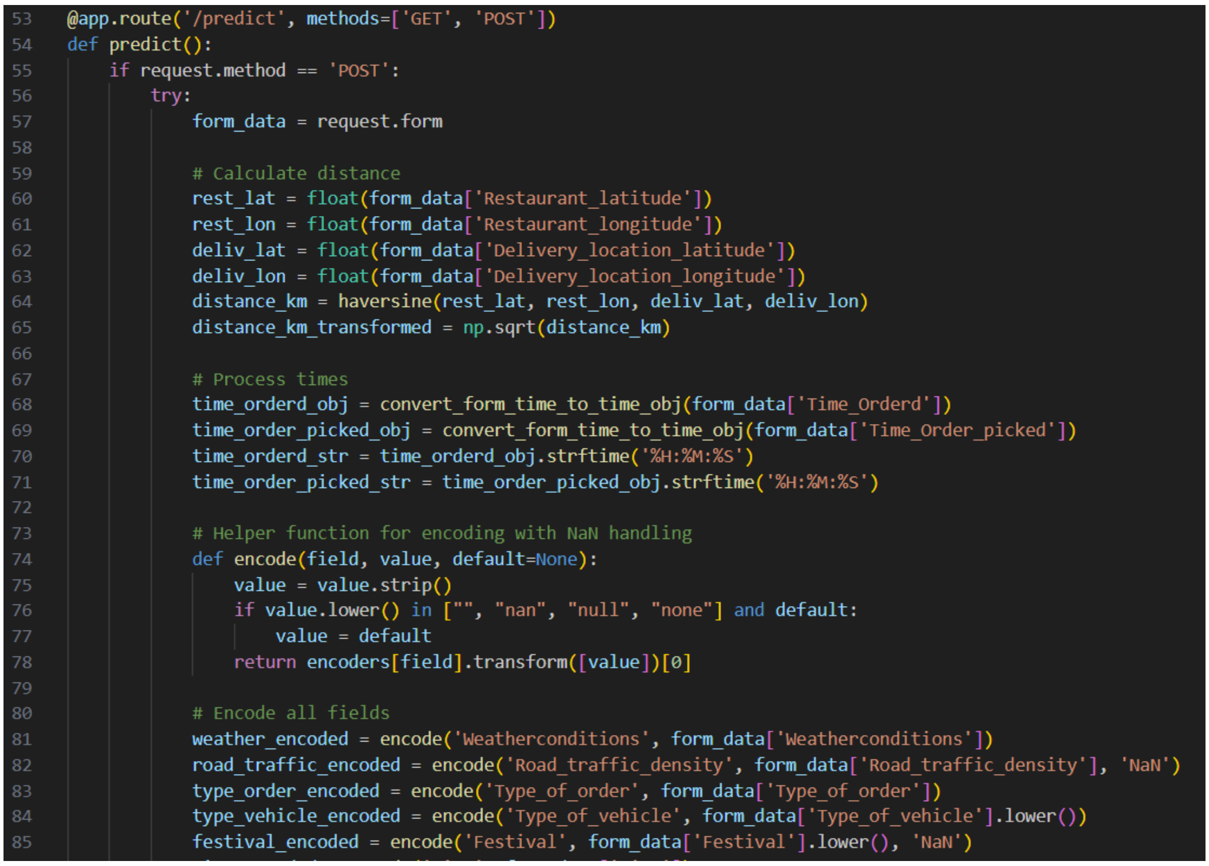
```python
@app.route('/about')
def about():
    return render_template('about.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

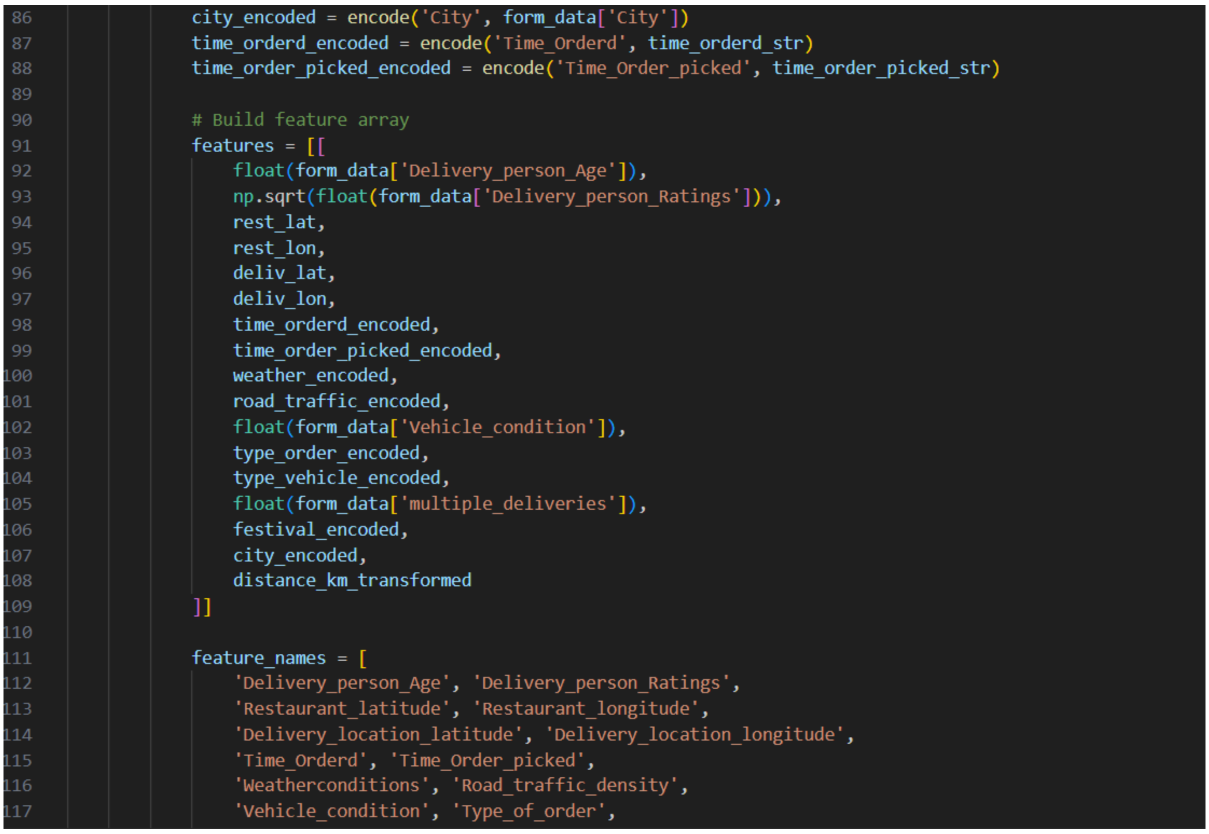
In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered.

Whenever you enter the values from the html page the values can be retrieved using POST Method.

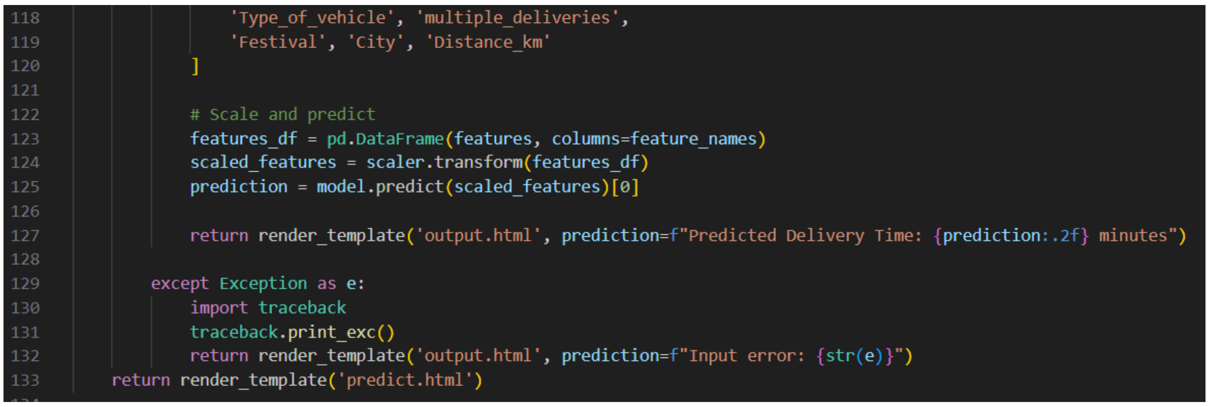
Retrieves the value from UI:



```python
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        try:
            form_data = request.form

            # Calculate distance
            rest_lat = float(form_data['Restaurant_latitude'])
            rest_lon = float(form_data['Restaurant_longitude'])
            deliv_lat = float(form_data['Delivery_location_latitude'])
            deliv_lon = float(form_data['Delivery_location_longitude'])
            distance_km = haversine(rest_lat, rest_lon, deliv_lat, deliv_lon)
            distance_km_transformed = np.sqrt(distance_km)

            # Process times
            time_orderd_obj = convert_form_time_to_time_obj(form_data['Time_Orderd'])
            time_order_picked_obj = convert_form_time_to_time_obj(form_data['Time_Order_picked'])
            time_orderd_str = time_orderd_obj.strftime('%H:%M:%S')
            time_order_picked_str = time_order_picked_obj.strftime('%H:%M:%S')

            # Helper function for encoding with NaN handling
            def encode(field, value, default=None):
                value = value.strip()
                if value.lower() in ["", "nan", "null", "none"] and default:
                    value = default
                return encoders[field].transform([value])[0]

            # Encode all fields
            weather_encoded = encode('Weatherconditions', form_data['Weatherconditions'])
            road_traffic_encoded = encode('Road_traffic_density', form_data['Road_traffic_density'], 'NaN')
            type_order_encoded = encode('Type_of_order', form_data['Type_of_order'])
            type_vehicle_encoded = encode('Type_of_vehicle', form_data['Type_of_vehicle'].lower())
            festival_encoded = encode('Festival', form_data['Festival'].lower(), 'NaN')
```



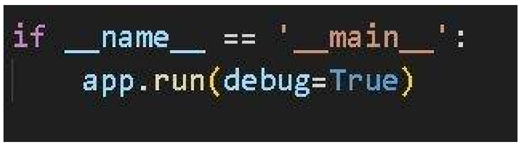
```python
            city_encoded = encode('City', form_data['City'])
            time_orderd_encoded = encode('Time_Orderd', time_orderd_str)
            time_order_picked_encoded = encode('Time_Order_picked', time_order_picked_str)

            # Build feature array
            features = [[
                float(form_data['Delivery_person_Age']),
                np.sqrt(float(form_data['Delivery_person_Ratings'])),
                rest_lat,
                rest_lon,
                deliv_lat,
                deliv_lon,
                time_orderd_encoded,
                time_order_picked_encoded,
                weather_encoded,
                road_traffic_encoded,
                float(form_data['Vehicle_condition']),
                type_order_encoded,
                type_vehicle_encoded,
                float(form_data['multiple_deliveries']),
                festival_encoded,
                city_encoded,
                distance_km_transformed
            ]]

            feature_names = [
                'Delivery_person_Age', 'Delivery_person_Ratings',
                'Restaurant_latitude', 'Restaurant_longitude',
                'Delivery_location_latitude', 'Delivery_location_longitude',
                'Time_Orderd', 'Time_Order_picked',
                'Weatherconditions', 'Road_traffic_density',
                'Vehicle_condition', 'Type_of_order',
```



```python
                'Type_of_vehicle', 'multiple_deliveries',
                'Festival', 'City', 'Distance_km'
            ]

            # Scale and predict
            features_df = pd.DataFrame(features, columns=feature_names)
            scaled_features = scaler.transform(features_df)
            prediction = model.predict(scaled_features)[0]

            return render_template('output.html', prediction=f"Predicted Delivery Time: {prediction:.2f} minutes")

        except Exception as e:
            import traceback
            traceback.print_exc()
            return render_template('output.html', prediction=f"Input error: {str(e)}")
    return render_template('predict.html')
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

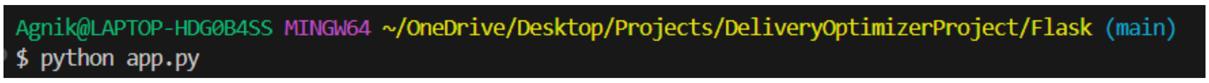
Main Function:



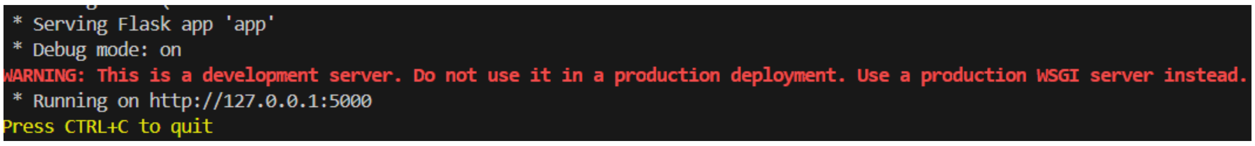
```python
if __name__ == '__main__':
    app.run(debug=True)
```

Activity 2.3: Run the web application

1. Open anaconda prompt from the start menu
2. Navigate to the folder where your python script is.
3. Now type "python app.py" command
4. Navigate to the localhost where you can view your web page.
5. Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

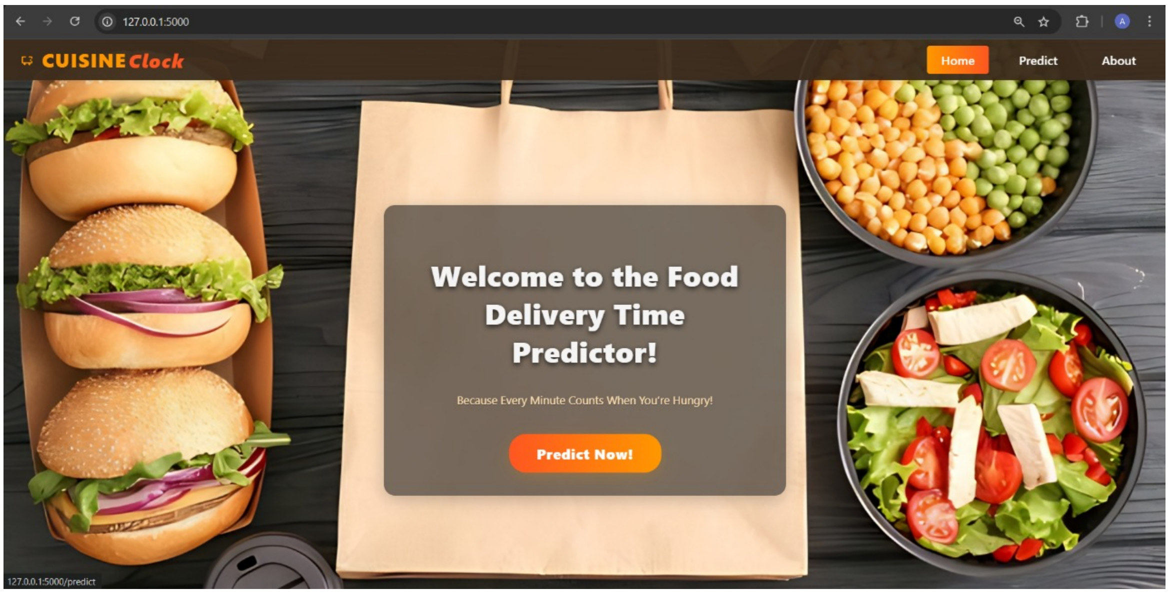


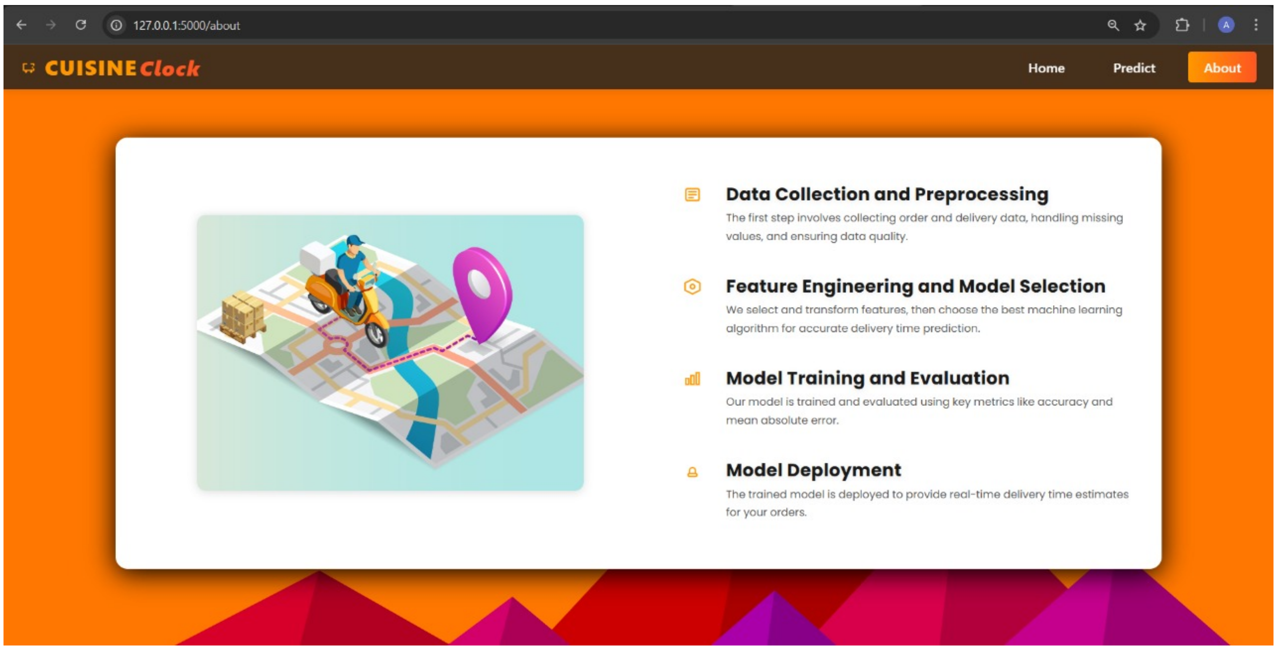
```
Agnik@LAPTOP-HDG0B4SS MINGW64 ~/OneDrive/Desktop/Projects/DeliveryOptimizerProject/Flask (main)
$ python app.py
```



```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result


127.0.0.1:5000
CUISINE Clock
Home
Predict
About
Welcome to the Food Delivery Time Predictor!
Because Every Minute Counts When You're Hungry!
Predict Now!
127.0.0.1:5000/predict



Data Collection and Preprocessing

The first step involves collecting order and delivery data, handling missing values, and ensuring data quality.

Feature Engineering and Model Selection

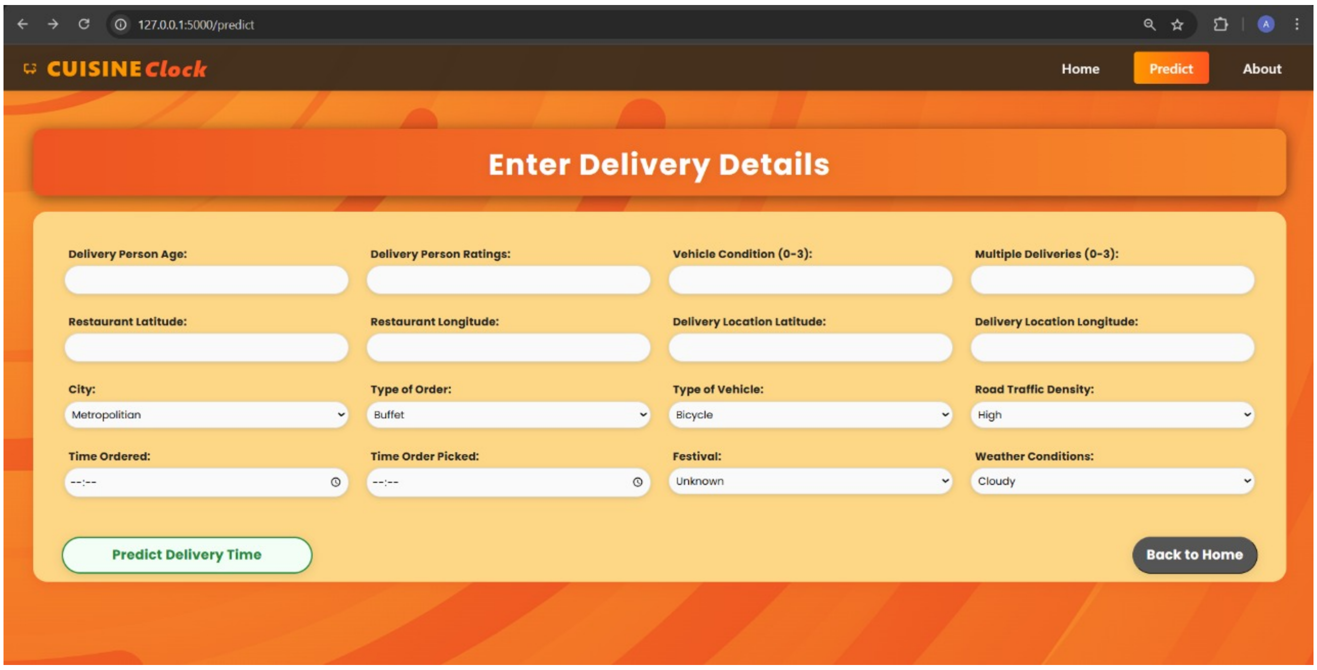
We select and transform features, then choose the best machine learning algorithm for accurate delivery time prediction.

Model Training and Evaluation

Our model is trained and evaluated using key metrics like accuracy and mean absolute error.

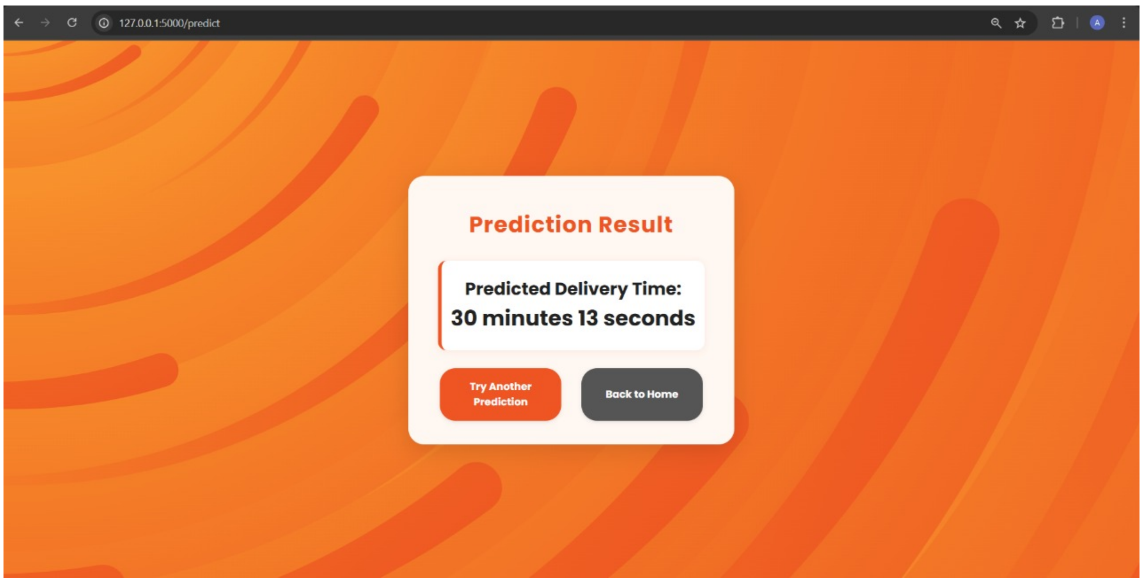
Model Deployment

The trained model is deployed to provide real-time delivery time estimates for your orders.



Enter Delivery Details

Delivery Person Age:	Delivery Person Ratings:	Vehicle Condition (0-3):	Multiple Deliveries (0-3):
Restaurant Latitude:	Restaurant Longitude:	Delivery Location Latitude:	Delivery Location Longitude:
City: Metropolitian	Type of Order: Buffet	Type of Vehicle: Bicycle	Road Traffic Density: High
Time Ordered: --:--	Time Order Picked: --:--	Festival: Unknown	Weather Conditions: Cloudy



Prediction Result

Predicted Delivery Time:
30 minutes 13 seconds

<u>Milestone 7: Project Demonstration & Documentation</u>

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided