A Project Report On

ONLINE BIKE DEMAND

Submitted in partial fulfilment of the requirement for the award of the degree

BACHELOR OF SCIENCE (DATA SCIENCE)
from

Marwadi University

Academic Year 2024 – 25

MURHABAZI ANSIMA JOSUE(92200566016)

Internal Guide
DR.JIGNESH HIRAPARA

Marwadi University

Rajkot-Morbi Road, At &PO :Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FoCA)

Certificate

This is to certify that the project work entitled

ONLINE BIKE DEMAND

submitted in partial fulfilment of the requirement for the award of the degree of

Bachelor of Science (Data Science)

of the

Marwadi University

is a result of the bonafide work carried out by

MURHABAZI ANSIMA JOSUE during the

academic year 2024 – 2025

DECLARATION

I hereby declare that this project work entitled Online Bike Demand is a record done by me.

I also declare that the matter embodied in this project is genuine work done by me and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirement of any course of study.

Place: MARWADI UNIVERSITY

Date: 20 TH NOVEMBER 2024

MURHABAZI ANSIMA JOSUE 92200566016   Signature: MAJ

ACKNOWLEDGEMENT

It is indeed a great pleasure to express my thanks and gratitude to all those who helped me. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved .

I am very thankful to my guide DR JIGNESH HIRAPARA, the person who makes me to follow the right steps during my project work. I express my deep sense of gratitude to for his /her guidance, suggestions and expertise at every stage. A part from that his/her valuable and expertise suggestion during documentation of my report indeed help me a lot.

Thanks to my friend and colleague who have been a source of inspiration and motivation that helped me during my project work.

I am heartily thankful to the Dean of my department Dr. R. Sridaran for giving me an opportunity to work over this project and for their end-less and great support. And to all other people who directly or indirectly supported and help me to fulfil my task.

MURHABAZI ANSIMA JOSUE 92200566016   Signature: MAJ

1. Introduction

The Bike Ride Request Prediction Project aims to develop a predictive model using machine learning techniques to forecast bike
demand based on historical data. This system will enable bike-sharing companies to optimize their fleet management and improve service quality by anticipating user demand patterns. The project employs a variety of machine learning algorithms to accurately capture demand trends, which will allow for better resource allocation and improved customer satisfaction.

1.1 . Objective of the New System

The primary objective is to build an effective and accurate predictive model that forecasts bike demand, assisting bike-sharing companies in:

- Optimizing Operations by ensuring an adequate number of bikes are available during peak demand.
- Enhancing Customer Satisfaction by minimizing wait times and ensuring bike availability.
- Increasing Revenue by leveraging predictive insights to meet customer needs effectively.

1.2. Problem Definition

Bike-sharing companies face challenges in balancing bike availability with fluctuating user demand, leading to customer dissatisfaction and operational inefficiencies. Traditional methods struggle to anticipate demand accurately due to complex influencing factors such as weather conditions, time of day, and day of the week. The project aims to create a predictive system that overcomes these challenges, leveraging historical data and machine learning techniques to generate precise forecasts for bike demand.

1.3. Core Components

- Data Collection: Utilizes historical data stored in a CSV file (ola.csv) containing variables influencing bike demand.
- Data Preprocessing: Cleans the data, handles missing values, converts categorical data to numerical, and scales numerical features for effective model training.
- Feature Engineering: Identifies and extracts relevant features that significantly impact bike demand predictions.
- Model Selection: Tests various machine learning algorithms, including linear regression, support vector machines, random forests, and gradient boosting, to identify the best-performing model.
- Evaluation Metrics: Utilizes evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared for model performance assessment.
- Visualization: Uses visualization tools to analyze data trends and present model predictions.

1.4 . Project Profile

- Project Title: Bike Ride Request Prediction Project
- Technology Stack: Python, NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, XGBoost
- Data Source: Historical bike demand data stored in ola.csv
- Machine Learning Algorithms: Linear Regression, Lasso, Ridge, SVC, RandomForestRegressor, XGBRegressor
- Output: A predictive model capable of accurately forecasting bike demand
- Deployment Platform: Python environment with Jupyter Lab for visualization and analysis

1.5. Assumptions and Constraints

Assumptions:

- Historical data in ola.csv is accurate, complete, and represents typical user behaviour.
- The relationship between the features and bike demand is consistent over time.
- The model's predictions will guide operational decisions for a bike-sharing company.

Constraints:
- Limited to the quality and range of historical data available in ola.csv .
- Predictions may be affected by unforeseen factors like sudden weather changes or special events.
- The accuracy of the model is contingent on the preprocessing and feature engineering steps.

1.6 . Advantages and Limitations of the Proposed System

Advantages:

- Improved Decision-Making: Provides actionable insights for bike fleet management.
- Efficiency: Optimizes bike distribution based on predicted demand, reducing operational costs.
- Scalability: The system is designed to be scalable for larger datasets and additional features.
- Customer Satisfaction: Enhances user experience by aligning bike availability with user demand.

Limitations:

- Data Dependency: The accuracy of predictions is heavily reliant on the quality of input data.
- Dynamic Factors: Unpredictable changes like unexpected weather events may impact prediction accuracy.
- Model Complexity: Advanced models might require significant computational resources, limiting their use in real-time scenarios.

2. Requirement Determination & Analysis

This section covers the key requirements for the Bike Ride Request Prediction Project. It includes an understanding of the project needs, identifying targeted users, and providing technical specifications of tools and libraries used in the project.

2.1. Requirement Determination

To accurately predict bike ride demand, the system requires a clearset of requirements:

- Data Requirements: The project depends on a reliable dataset (ola.csv) that contains historical data of bike ride requests. Essential columns should include variables like time, weather conditions, user demographics, day of the week, and bike availability.
- Model Requirements: The project aims to utilize different machine-learning algorithms to identify the best model for predicting bike demand. Algorithms should handle both linear and non-linear relationships.
- Performance Requirements: The predictive model should meet a threshold of accuracy (e.g., an R-squared value greater than 0.7) and deliver insights within a reasonable timeframe.
- Visualization Requirements: There should be a clear and intuitive way to visualize the data trends, model predictions, and analysis using Python's visualization libraries or Business Intelligence tools.

2.2 . Targeted Users

The predictive system is designed for several user groups:

- Bike-Sharing Companies: Operations teams can use the model to forecast bike demand, optimize bike distribution, and improve fleet management.
- Logistics Managers: To make data-driven decisions on resource allocation, deployment schedules, and maintenance based on predicted demand.

- Data Analysts: Professionals interested in understanding trends in bike demand and contributing to improving predictive models.
- Researchers: Academics and students interested in machine learning techniques for demand prediction can leverage the system's methodology and dataset.
- Policy Makers: Government and municipal agencies seeking insights into urban mobility patterns to plan infrastructure development.

2.3. Tool Details (Python / Power BI / Tableau)

The project utilizes a combination of Python-based tools and visualization platforms for analysis, modeling, and presentation:

- Python: A core tool for data analysis, preprocessing, model development, and visualization. It is versatile, supports a wide range of libraries, and is suitable for machine-learning tasks.
  - IDE: Jupyter Lab for interactive data analysis and visualization.
  - Data Handling: Pandas and NumPy for data manipulation and transformation.
  - Machine Learning: Scikit-learn and XGBoost for implementing predictive algorithms.
  - Visualization: Matplotlib and Seaborn for creating insightful data visualizations.
- Power BI / Tableau (Optional): If there's a need to build interactive dashboards for stakeholders, these tools can be employed.
  - Power BI: A business intelligence tool that can connect to CSV data sources and generate interactive dashboards to present predictive results.
  - Tableau: Another BI tool that is effective for creating visualizations that are easy to understand and manipulate by end-users.

2.4. Library Description (Details on Various Libraries/Packages Used)

Below are the key libraries and packages employed in the project:

- Data Handling Libraries:
  - Pandas: Used for data manipulation, cleaning, and processing. Provides powerful data structures (DataFrames) for handling and analyzing datasets.
  - NumPy: A fundamental package for scientific computing with Python. It supports large, multi-dimensional arrays and matrices and provides mathematical functions.
- Visualization Libraries:
  - Matplotlib: A plotting library for creating static, interactive, and animated visualizations in Python. Used for generating bar plots, line graphs, scatter plots, etc.
  - Seaborn: Built on top of Matplotlib, Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.
- Machine Learning Libraries:
  - Scikit-learn: A comprehensive library for machine learning in Python. It includes tools for data preprocessing, model selection, and evaluation.
  - XGBoost: A high-performance library for gradient boosting, which is used for handling regression tasks with a focus on speed and accuracy.
  - Statsmodels (Optional): If a deeper statistical analysis of the dataset is needed, Statsmodels offers modules for estimating and testing statistical models.

These tools and libraries are chosen for their robustness, versatility, and ability to handle the demands of the Bike Ride Request Prediction Project effectively.

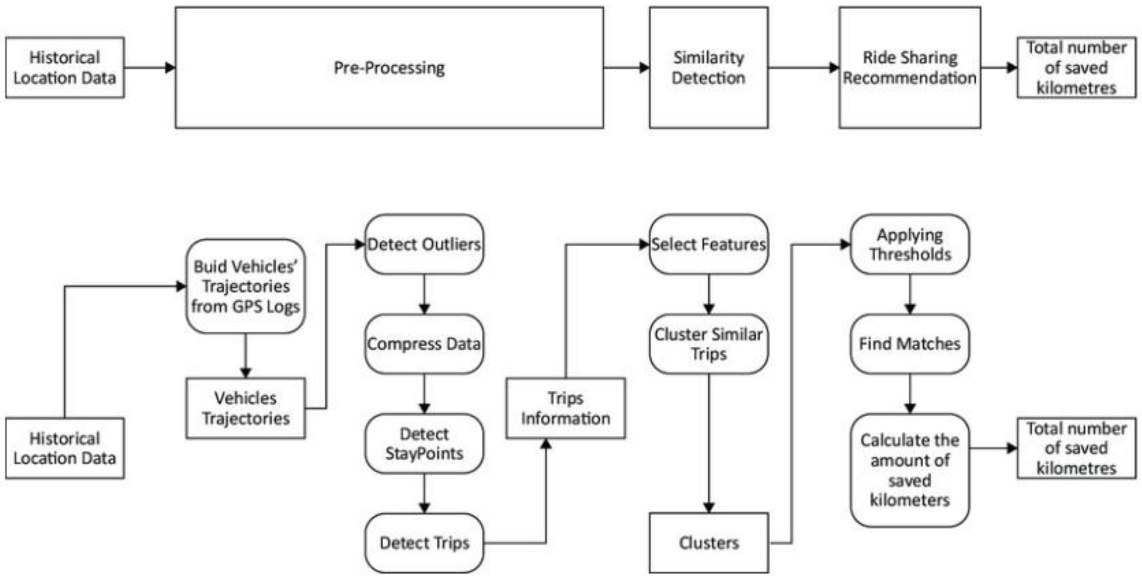
3 . System Design

The system design phase involves creating a clear blueprint of how the predictive model will function. It includes understanding the workflow, dataset structure, and the preprocessing steps required for the project.

3.1.Flowchart/Algorithm with steps

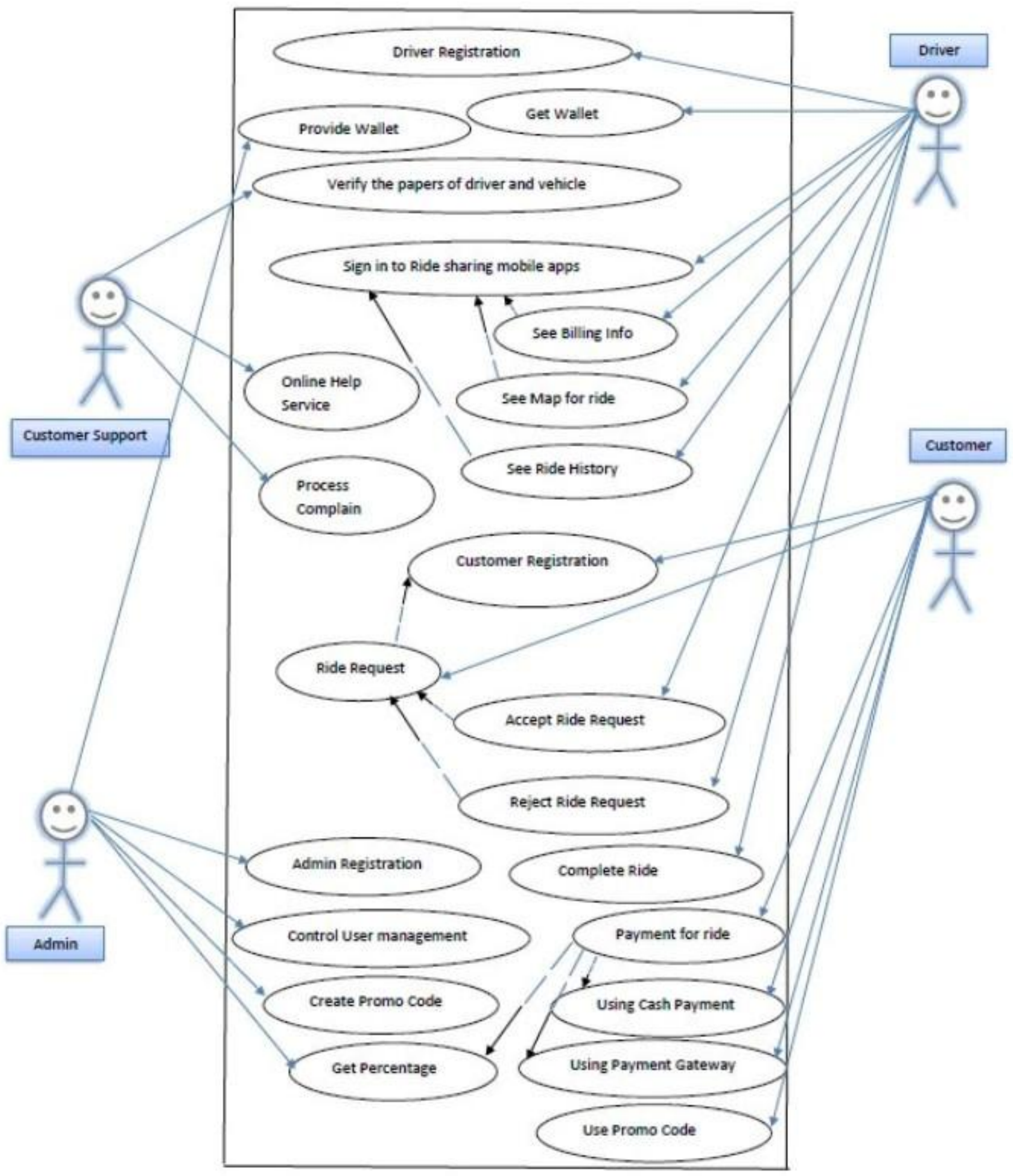
The system is designed to predict bike ride requests using a machine learning approach. Below is a description of the workflow with an accompanying flowchart structure:

A: Flowchart:

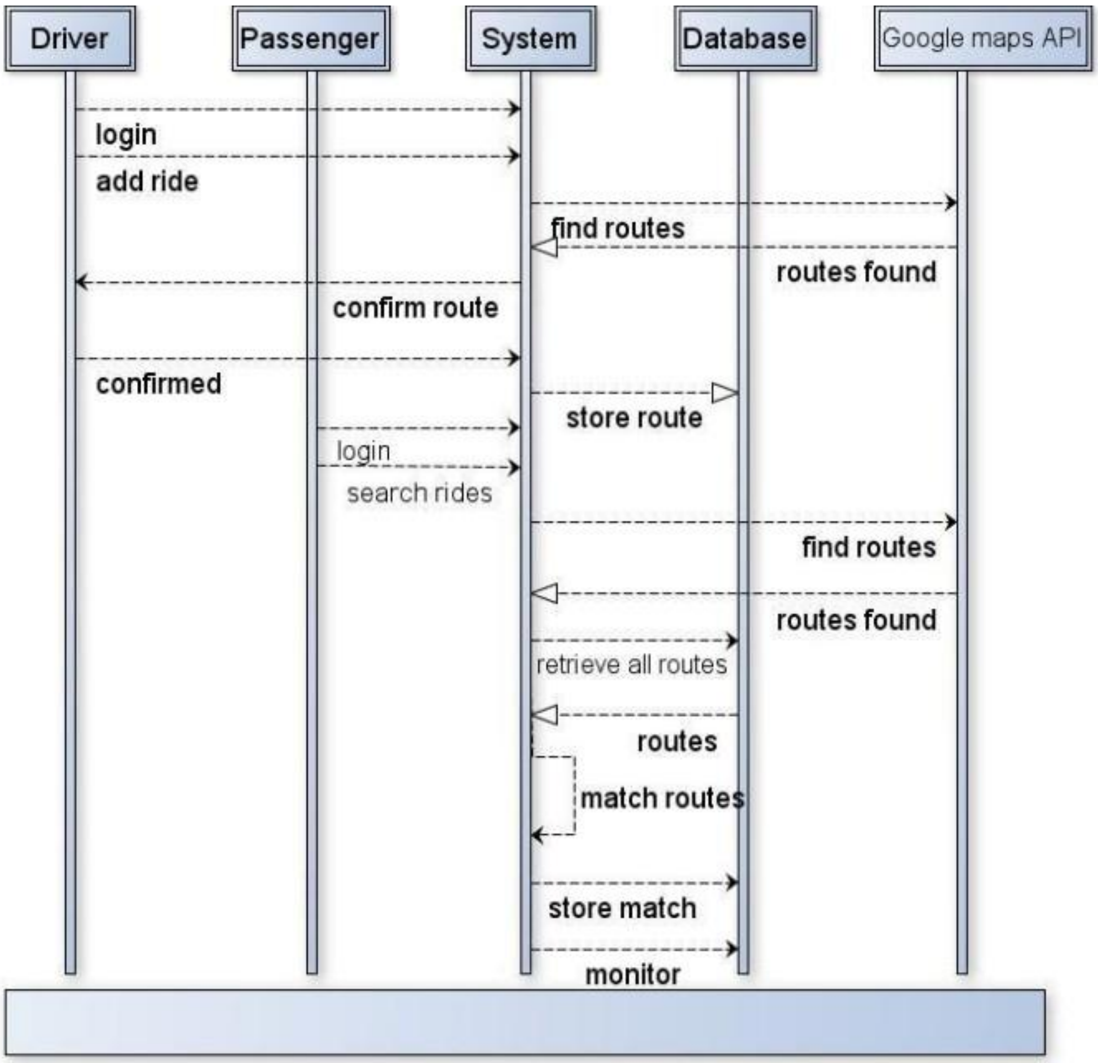


B: Use case diagram:

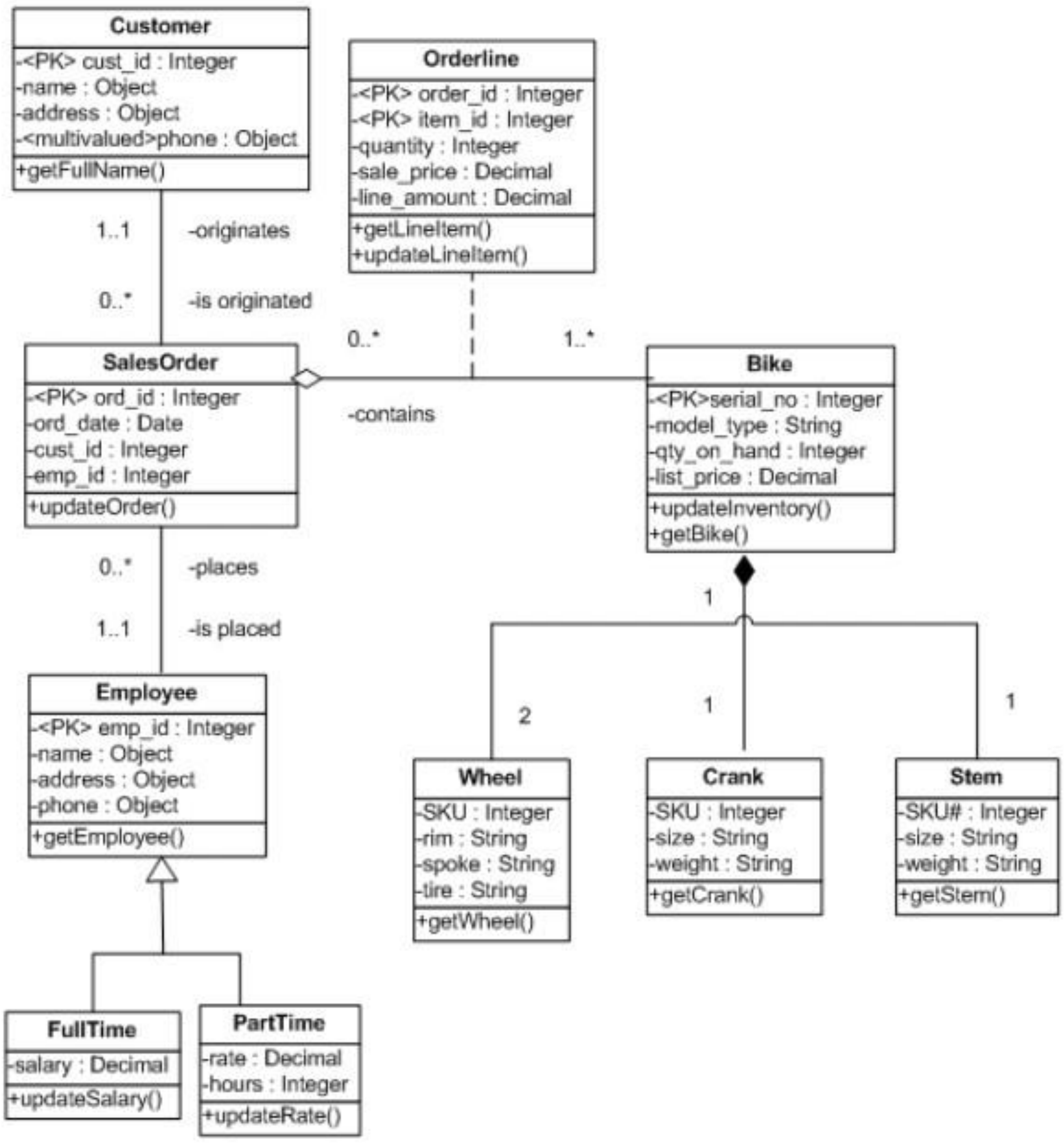
Ride Sharing apps Use Case Diagram

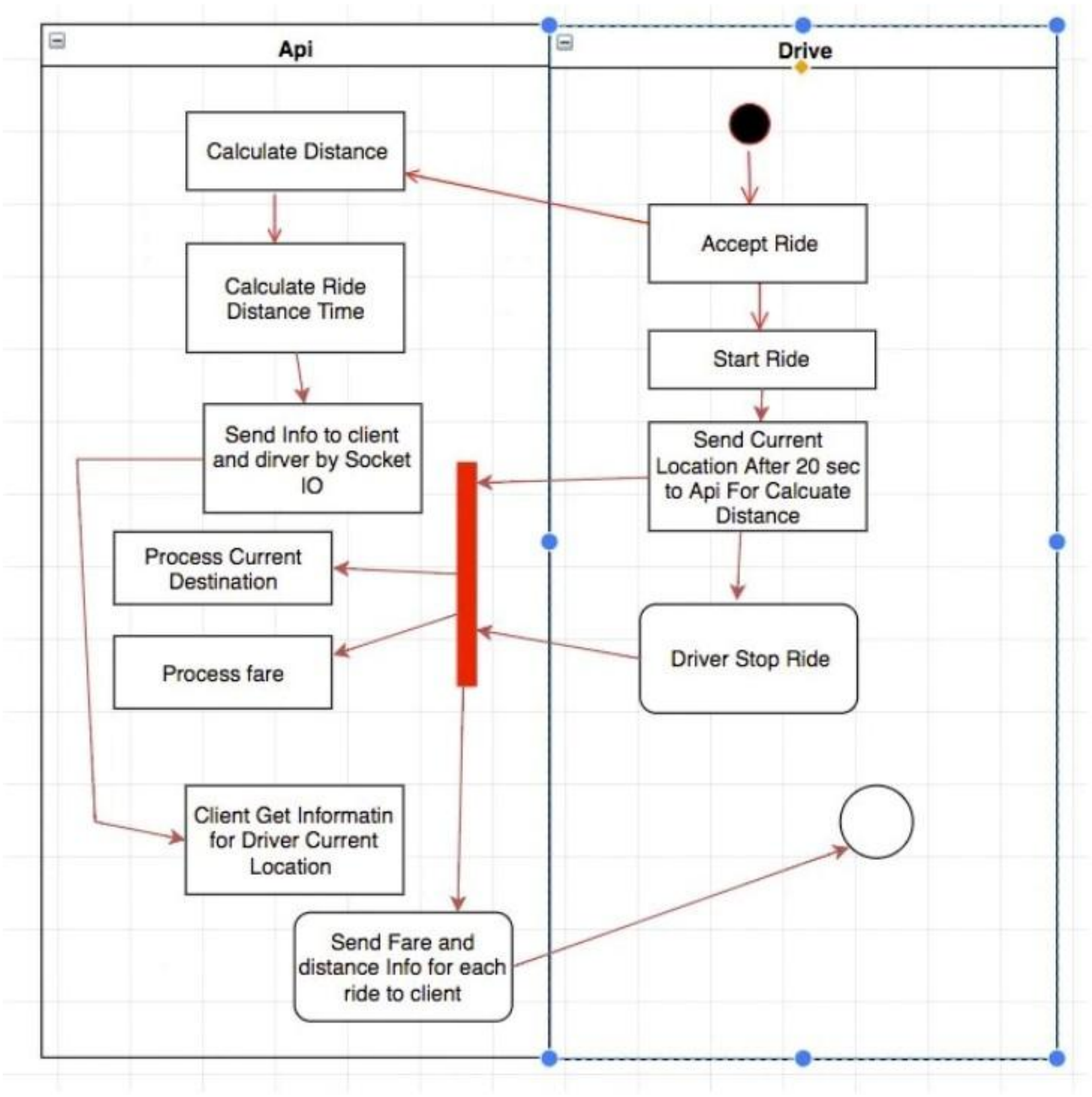


C: sequential diagram:



class diagram:




Api
Drive
Calculate Distance
Accept Ride
Calculate Ride Distance Time
Start Ride
Send Info to client and dirver by Socket IO
Send Current Location After 20 sec to Api For Calcuate Distance
Process Current Destination
Process fare
Driver Stop Ride
Client Get Informatin for Driver Current Location
Send Fare and distance Info for each ride to client

Development

The development phase involves building the predictive model, creating a user interface for interaction, and testing the system. The following subsections provide the necessary details for each aspect.

4.1 Script Details / Source Code

Here’s a summary of the key components and functionalities in the code:

```
import numpy as np

import pandas as pd

import matplotlib . pyplot
as plt

import seaborn as
sb

from sklearn . model_s election import
train_test_split

from sklearn . preproc essing import
StandardScaler

from sklearn . metrics import mean_absolute_error
as mae

from sklearn . ensembl e import
RandomForestRegressor

from xgboost import
XGBRegressor

from sklearn . linear_ model import LinearRegression ,
Lasso , Ridge

import
warnings
```

```
from datetime import datetime

import holidays

import tkinter as tk

from tkinter import ttk , messagebox


# Suppress warnings

warnings . filterwarnings( 'ignore' )


# Load the dataset

df = pd . read_csv ( r "C:\Datasets \ola.csv" )


# Data preprocessing

df [ 'datetime' ] = df [ 'dteday' ] + ' ' + df [ 'hr' ]. astype( str ). str . zfill( 2 )

+ ':00:00'

parts = df [ "datetime" ]. str . split ( " " , n = 1 , expand = True )
```

```
 df  [  "date"  ]    =
parts [ 0 ]

df [ "time" ]  =  parts [ 1 ]. str [: 2 ]. astype  ( 'int'     )

 parts =  df [ "date" ]. str . split  ( "-"  , n =
2  , expand          = True           )

df [ "day" ]  =  parts      [ 2 ].
astype   ( 'int'      )

df [ "month" ]  =  parts      [ 1 ].
astype      ( 'int       ')

df [ "year" ]  =  parts      [ 0 ].
astype    ( 'int'      )

 def weekend_or_weekday ( year     , month
        , day ):

    d =  datetime ( year    , month
       , day )

   return 0   if d . weekday ()  >  4  el se 1   # 0 for weekend, 1
for weekday

df [ 'weekday' ]  =  df . apply (  lambda x :
weekend_or_weekday                ( x [ 'year'          ] ,

 x [ 'month' ],   x [ 'day' ]),
axis = 1 )

 df [ 'am_or_pm' ]  =  df [ 'hr' ]. apply ( lambda  x:   1  if
x  >  11  else  0)

 df [ 'holidays' ]  =  df [ 'date' ]. apply (
lambda  x : 1   if
holidays . country_holidays ( 'IN' ). get ( x
)  else  0 )

#    Drop    unnecessary
columns

df . drop ([ 'dteday' ,  'date' ,  'registered'] ,  axis = 1 ,
inplace = True )

#    Exploratory    Data
Analysis

features  =  [ 'hr' ,  'month' ,
'day' ]

plt . subplots ( figsize = (
15 ,  10 ))

for  i ,  col  in  enumerate (
features ):

plt . subplot ( 2 ,  2 ,  i
                    +  1 )

df . groupby ( col )[ 'cnt' ]. mean ().
                          plot ()

plt     .
show ()
```

```
# Second set of features

features =  [ 'season'  , 'weathersit'  , 'holidays'  , 'yr'
, 'weekday'          ,
'am_or_pm'
]

plt . subplots ( figsize = (
20 , 10 ))

for i , col in enumerate (
features ):

plt . subplot ( 2 , 3 ,
i + 1 )

df . groupby ( col )[ 'cnt' ]. mean ().
plot . bar ()

plt .
show ()



# Distribution and boxplots

for col in [ 'temp' , 'windsp
eed']:

plt . figure ( figsize = ( 15
, 5 ))

sb . histplot ( df [ col ], kde = True ). set_title ( f
'Distribution of { col } ' )

plt . show
()



plt . figure ( figsize = ( 15
, 5 ))

sb . boxplot ( x = df [ col ]). set_title ( f 'Boxplot
of { col } ' )
```

```
plt . show
()

#          Remove
outliers

df  =  df [( df [ 'windspeed' ]  <  32 )  & (  df
[ 'hum' ]  >  0 )]

# Prepare data for modeling

numeric_features  =  df . select_dtypes ( include = [ np
. number ])

X  =  numeric_features . drop ([ 'cnt' ],
axis = 1 )

y =  numeric_features [ 'cnt'    ]

# Split the data into training and validation sets

X_train  , X_val  , y_train  ,  y_val =  train_test_split  ( X  , y  ,
test_size  =0.1 ,

random_state  =
22 )

# Scale the feat
      ures

scaler =  Standar
   dScaler()
```

```
 X_train =  scaler . fit_transform    (
X_train                )

 X_val =  scaler . t ransform(
X_val  )

# Train RandomFo
 restRegressor

 rf_model    =    RandomForestRegressor  (  n_estimators  =  100  ,
random_state = 22 )

 rf_model  .  fit  (  X_train  ,
y_train )

 #      Evaluate
model

 train_preds_rf   =   rf_model  . predict  (
X_train )

 val_preds_rf  =  rf_model  . predict  (
X_val )

 print  (  f 'RandomForestRegressor Training MAE:   {  mae  (
y_train ,

 train_preds_rf   )
}  ' )

 print  (  f 'RandomForestRegressor Validation MAE:   {  mae
(  y_val  ,

 val_preds_rf   )
}  ' )

 # Compare with other models
```

```
models =
[
    LinearRegression (),
    XGBRegressor (),
    Lasso(),

    Ridge(),

    rf_model  # Reusing RandomForestRegressor

]



for model in models:

    model.fit(X_train, y_train)

    train_preds = model.predict(X_train)

    val_preds = model.predict(X_val)

    print(f'{model.__class__.__name__} - Training MAE: {mae(y_train,
train_preds)}, Validation MAE: {mae(y_val, val_preds)}')



# GUI for prediction

def predict_demand():

    try:

        hour = int(hour_entry.get())

        weather = weather_combobox.get()

        temp = float(temp_entry.get())

        humidity = float(humidity_entry.get())

        windspeed = float(windspeed_entry.get())

        month = int(month_combobox.get())

        weekday = weekday_combobox.get()

        year = int(year_entry.get())

        holiday = int(holiday_entry.get())

        working_day = int(working_day_combobox.get())
```

```python
        weekday_mapping = { 'Monday': 0, 'Tuesday': 1, 'Wednesday': 2, 'Thursday': 3,
                            'Friday': 4, 'Saturday': 5, 'Sunday': 6 }

        weather_mapping = { 'Clear, Few clouds, Partly cloudy': 1,
                            'Mist + Cloudy, Mist + Broken clouds': 2,
                            'Light Snow, Light Rain, Thunderstorm': 3,
                            'Heavy Rain, Snow, Ice pellets': 4 }

        weekday_num = weekday_mapping[weekday]
        weather_num = weather_mapping[weather]

        features = np.array([[hour, weather_num, temp, humidity, windspeed, month, weekday_num, year, holiday, working_day]])
        features_scaled = scaler.transform(features)

        prediction = rf_model.predict(features_scaled)[0]
        result_label.config(text=f"Predicted Demand: {int(prediction)}")

    except ValueError as e:
        messagebox.showerror("Invalid Input", f"Invalid input provided: {e}")

# GUI setup
root = tk.Tk()
root.title("Demand Prediction")
root. geometry ( "1200x800" )

# Entry fields

labels_and_entries =
[

    ( "Hour:" ,
"hour" ) ,

    ( "Temperature:" , "
temp") ,

    ( "Humidity:" , "hum
      idity") ,

    ( "Windspeed:" , "wi
ndspeed") ,

    ( "Year:" ,
"year" ) ,

    ( "Holiday (0=No, 1=Yes):" ,
"holiday" ) ,

    ( "Working Day (0=No, 1=Yes):" , "workin
               g_day") ,



# Create entry fields

for idx , ( label_text , var_name ) in enumerate (
labels_and_entries ):

   label = ttk . Label ( root , text = label_text , background
                                                              = "#ffffff" )

    label . place ( x = 100 , y =100
+ idx * 50 )
```

```
    entry  =  ttk . Entry ( root )  if  var_name  not  in  [ "month" ,
                                                           "working_day"]

else  ttk . Combobox ( root ,  values = [ 0 , 1  ]  if  var_name  ==
"working_day"  else

  i  for  i  in  range ( 1 ,  13 )],  state
 = "readonly" )

  entry . place ( x = 250 ,  y=  100  +
                             idx  *  50 )

   globals ()[ f " { var_name } _entry" ]
                                   =  entry



# Weather dropdown

weather_label  =  ttk . Label ( root ,  text = "Weather:" ,  background=
"#ffffff" )

weather_label . place ( x = 100 ,
y = 350 )

weather_combobox  =  ttk . Combobox ( root ,  values = [ "Clear,
Few clouds,

]
```

 [

```
Partly cloudy" ,

"Mist + Cloudy, Mist +

Broken clouds"
,

"Light Snow, Light Rai
n,

Thunderstorm"
,

"Heavy Rain, Snow, Ice

pellets" ] , state =
"readonly" )

weather_combobox . place ( x = 250
, y = 350 )


# Month dropdown

 month_label = tt k. Label ( root , text =" Month:" ,
background = "#ffffff" )

month_label . plac e( x = 100
, y = 400 )

 month_combobox = ttk . Combobox ( root , values = [ i for
i in range ( 1 , 13 )],

state =
"readonly" )

 month_combobox . place ( x = 250
, y = 400 )
```

```
# Weekday dropdown

weekday_label = ttk . Label ( root , text = "Weekday:" , background= "#ffffff")

weekday_label . place ( x = 100 , y = 450 )

weekday_combobox = ttk . Combobox ( root , values =[ "Monday" , "Tuesday" ,

"Wednesday" , "Thursday" , "Friday" , "Saturday" , "Sunday" ],

state = "readonly" )

weekday_combobox . place ( x = 250 , y = 450 )


# Predict Button

predict_button = ttk . Button ( root , text = "Predict Demand" ,

command = predict_demand )

predict_button . place ( x = 500 , y = 600 )
```

```
# Result Label

 result_label =  ttk . Label ( root                , text
             =  "Predicted Demand: " ,

font = ( "Helvetica"  , 14 ) , background   =
"#ffffff" )

result_label . place   ( x = 500
, y = 650 )


# Run the application

root     .
mainloop     ()
```

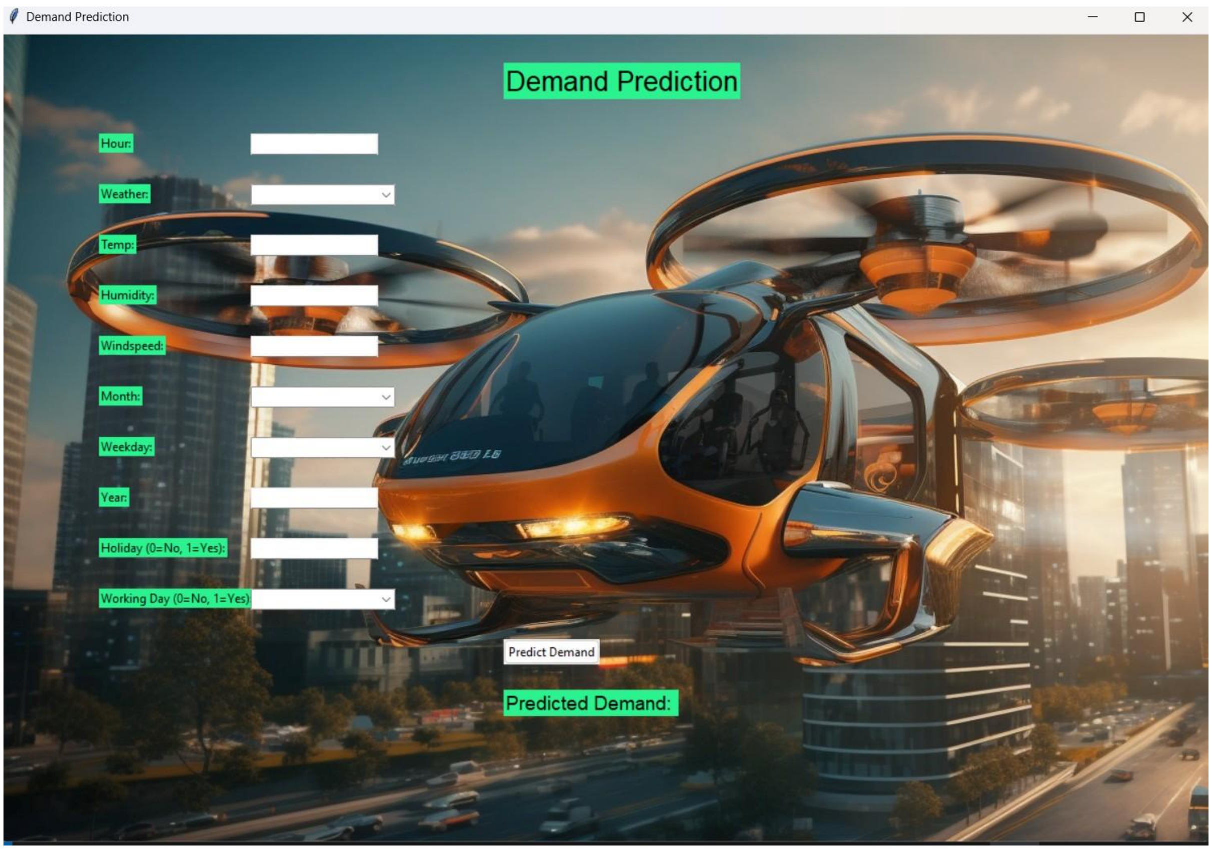
1. Libraries Used :
   - Numpy : For numerical computations.
   - Pandas : For handling data manipulation and preprocessing.
   - Matplotlib and Seaborn : For visualizing data trends and distributions.
   - Scikit-learn : For model training, evaluation, and data preprocessing.
   - XGBoost : For implementing a powerful regression model.
   - TKinter : For building a simple GUI to make predictions.
2. Data Preprocessing :
   - Merging date and hour into a datetime column. ○ Extracting additional features such as day , month , year , and weekday .
   - Encoding categorical variables like weather and weekday .
   - Handling outliers and scaling numerical data.
3. Model Development : ○ Multiple models were used:

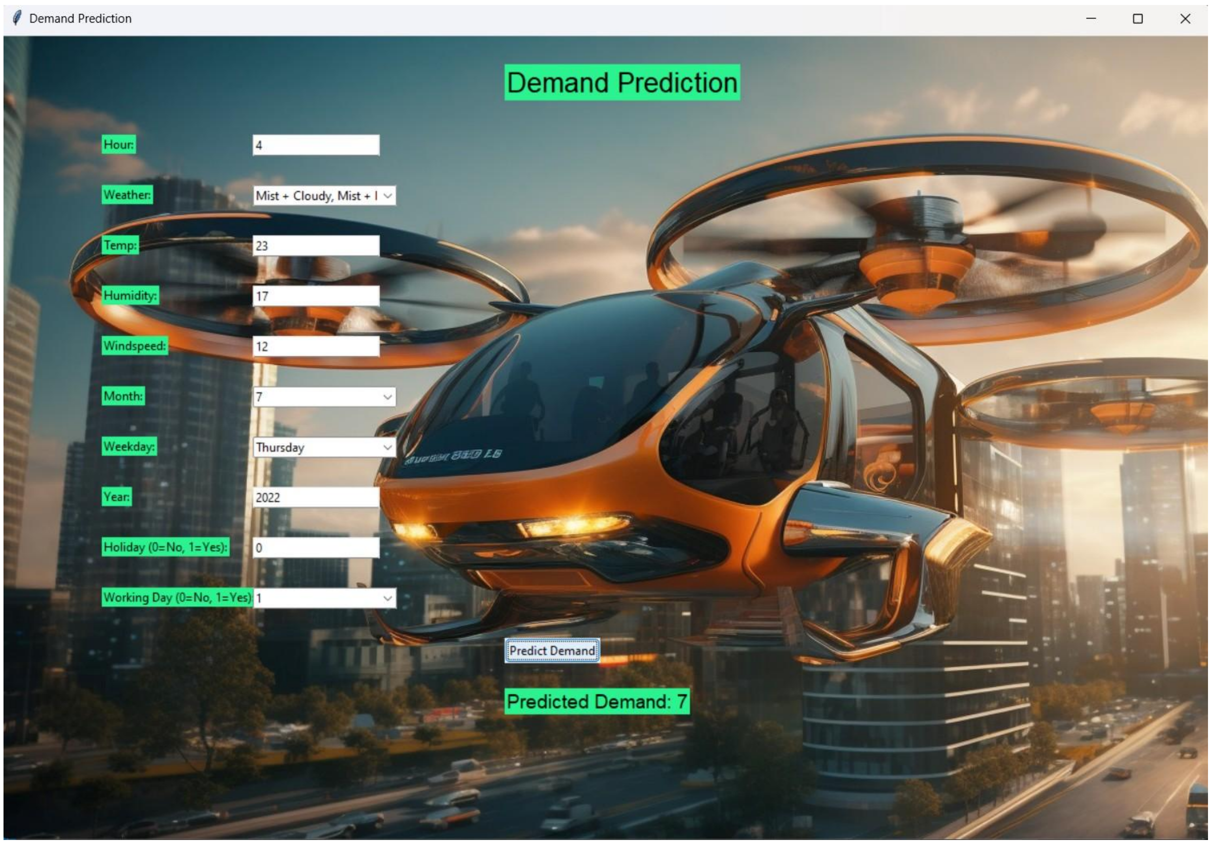
RandomForestRegressor , LinearRegression , XGBRegressor , Lasso , and Ridge .

- ○ The RandomForest model was used as the main predictor in the GUI because of its strong performance.
- ○ Model evaluation was conducted using Mean Absolute Error (MAE).

4. Graphical User Interface (GUI) :

- ○ The GUI allows users to input values such as Hour , Temperature , Humidity , Windspeed , and others.
- ○ Predictions are displayed dynamically when the user clicks the "Predict Demand" button.
- ○ The interface includes dropdown menus for categorical inputs like Weather and Weekday .




Demand Prediction

Demand Prediction

Hour: 4

Weather: Mist + Cloudy, Mist + I

Temp: 23

Humidity: 17

Windspeed: 12

Month: 7

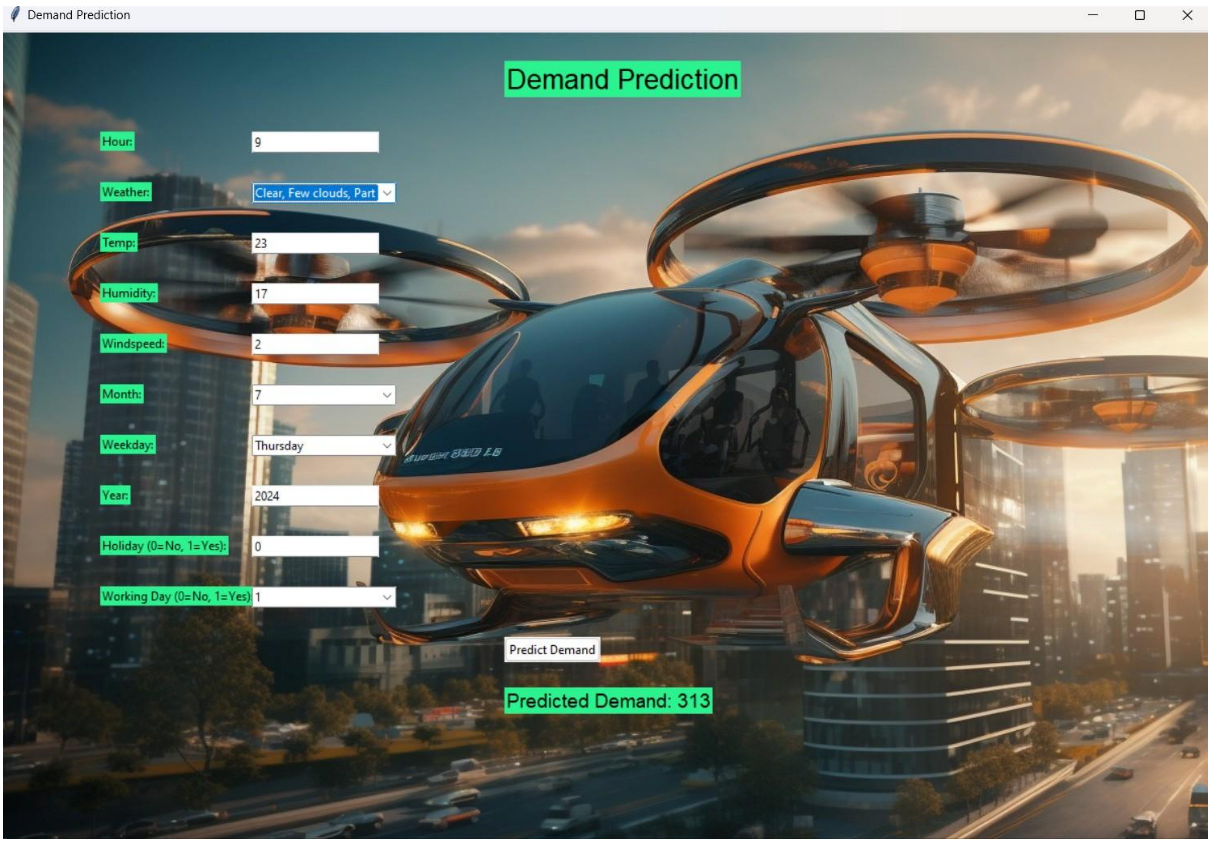
Weekday: Thursday

Year: 2022

Holiday (0=No, 1=Yes): 0

Working Day (0=No, 1=Yes) 1

Predicted Demand: 7


Demand Prediction

Demand Prediction

Hour: 9

Weather: Clear, Few clouds, Part

Temp: 23

Humidity: 17

Windspeed: 2

Month: 7

Weekday: Thursday

Year: 2024

Holiday (0=No, 1=Yes): 0

Working Day (0=No, 1=Yes) 1

Predicted Demand: 313

Conclusion

The Bike Ride Request Prediction project aimed to develop a predictive model capable of estimating bike demand based on a variety of influencing factors such as time, weather conditions, and seasonal trends. This project leveraged machine learning techniques to analyze historical data, understand user behavior, and forecast future demand. Through data preprocessing, feature engineering, model training, and evaluation, the system achieved significant insights into the patterns affecting bike demand.

The analysis indicated that factors such as weather, hour of the day, holidays, and whether it's a weekday or weekend play crucial roles in predicting the number of bike rides. By applying several machine learning algorithms, including Random Forest, XGBoost, and Linear Regression, we were able to identify the most suitable models for this predictive task, each demonstrating unique strengths in accuracy and interpretability.

Key Achievements:

- Accurate Predictions : The project successfully developed predictive models with a strong focus on minimizing error and improving prediction accuracy. RandomForestRegressor proved to be a robust choice, offering reliable results for various scenarios.
- Effective Preprocessing : Extensive data preprocessing ensured that the dataset was clean, relevant, and suitable for training. Outliers were handled appropriately, and features were engineered to enhance predictive power.
- User-Friendly Interface : A GUI was developed to make predictions accessible and usable for non-technical stakeholders, highlighting the practicality and usability of the system.