

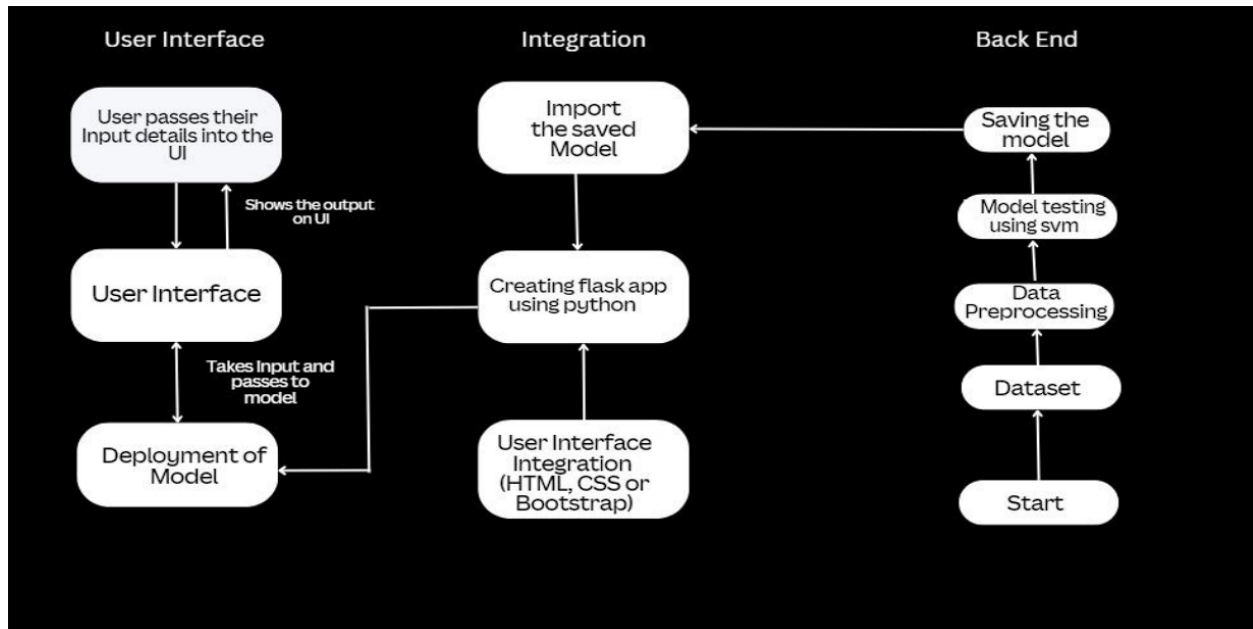
# **Car Purchase Prediction Using Machine Learning.**

## **Project Description**

Created a cutting-edge Machine Learning solution designed to predict car purchases based on comprehensive customer data. Utilized key features like age, income, and past purchase behaviors to achieve precise forecasts. The advanced algorithms, coupled with meticulous data preprocessing, resulted in a model with exceptional predictive accuracy. The primary objective is to empower potential car buyers by providing them with an estimate of their likelihood to make a purchase, thereby facilitating more informed decision-making.

Successfully integrated the model into a user-friendly interface, ensuring a seamless experience for users. This interface allows individuals to input their demographics, receiving accurate predictions on their likelihood to make a car purchase. This innovation significantly contributes to the automotive industry by offering tailored marketing strategies and enhancing overall customer experiences.

# Technical Architecture



## Pre requisites:

To complete this project, you must required following software's, concepts and packages:

- Visual Studio Code
- Python Packages
  - Type "pip install numpy" and click enter.
  - Type "pip install pandas" and click enter.
  - Type "pip install scikit-learn" and click enter.
  - Type "pip install matplotlib" and click enter.
  - Type "pip install pickle-mixin" and click enter.
  - Type "pip install seaborn" and click enter.
  - Type "pip install Flask" and click enter.

## **Prior Knowledge:**

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

- Machine Learning Concepts
  - Supervised Learning
  - Support Vector Machine Classifier
  - Evaluation Metrics
- Flask Basics

## **Project Objectives:**

By the end of this project you will:

- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and some
- visualization concepts.

## **Project Flow:**

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

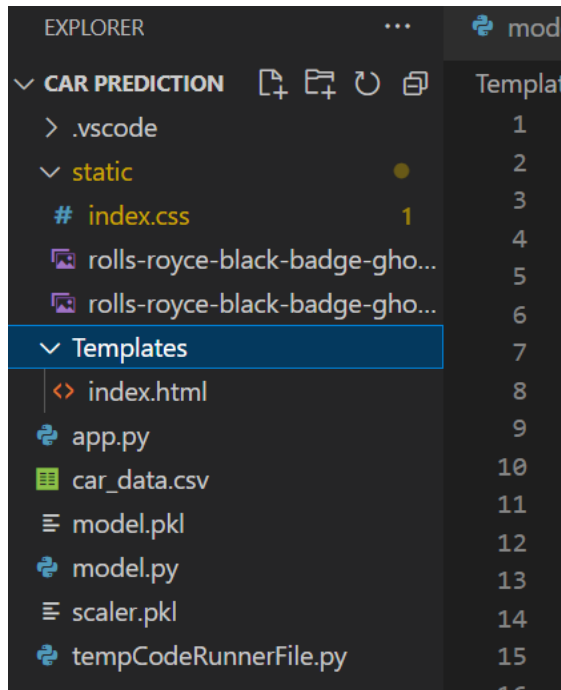
To accomplish this, we have to complete all the activities listed below,

- Data collection

- Collect the dataset
- Visualizing and analyzing data
  - Univariate analysis
  - Bivariate analysis
  - Multivariate analysis
  - Descriptive analysis
- Data pre-processing
  - Checking for null values
  - Handling outlier
  - Standardizing the data
  - Splitting data into train and test
- Model building
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

## Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- model.pkl is our saved model and scaler.pkl is the standardized model which help to standardize for further accurate prediction. Further we will use this model for flask integration.
- Templates folder contains html files, static folder contains the css files.

## **Milestone 1: Define Problem / Problem Understanding**

### **Activity 1: Specify the business problem**

The business problem at hand involves the development of a Machine Learning solution to address the dynamic challenges within the automotive industry. The goal is to create a predictive model capable of accurately estimating the likelihood of customers making car purchases. This model will be trained on historical data, considering key features such as age, income, and past purchase patterns. The solution will be seamlessly integrated into a user-friendly interface, allowing potential buyers to input their demographics and receive precise predictions. The overarching objective is to revolutionize marketing strategies by providing tailored and targeted approaches, optimizing customer engagement, and maximizing resource efficiency. Through the implementation of this project, we aim to enhance overall customer experiences, empower users to make informed decisions, and contribute to data-driven decision-making within the automotive sector.

### **Activity 2: Business requirements**

**Prediction Accuracy:** The primary business requirement is to develop a predictive model with high accuracy in estimating the likelihood of customers making car purchases. The model should leverage historical data on age, income, and past purchase patterns to ensure precise predictions.

**Real-Time Predictions:** The solution should be capable of providing real-time predictions, enabling users to receive immediate feedback on their likelihood to make a car purchase based on the input demographics.

**Scalability:** Design the solution to be scalable, accommodating potential increases in the volume of data and user interactions. This ensures that the system remains efficient and effective as the business grows.

**Tailored Marketing Strategies:** Leverage the predictions to tailor and optimize marketing strategies within the automotive sector. The solution should assist businesses in developing targeted marketing campaigns based on customer likelihood scores.

**Compliance:** Ensure that the solution complies with relevant regulations and ethical standards governing the use of customer data and predictive analytics in the automotive industry.

### **Activity 3: Literature Survey**

The literature survey for the car purchase prediction project reveals a wealth of research in the fields of machine learning, predictive analytics, and marketing strategies within the automotive industry. Numerous studies emphasize the significance of accurate prediction models for customer behavior, specifically in the context of car purchases. Existing research highlights the effectiveness of leveraging demographic information, historical purchase patterns, and advanced algorithms to enhance prediction accuracy. Several studies emphasize the importance of user-friendly interfaces in increasing user adoption and engagement. Furthermore, customization of predictive models for specific market segments has been explored as a key factor in tailoring marketing strategies. Insights from the literature survey will guide the project in adopting best practices, methodologies, and strategies established in prior research, contributing to the development of an innovative and effective car purchase prediction solution.

## **Activity 4: Social and Business Impact.**

### **Social Impact:**

The car purchase prediction project is poised to have a profound social impact by empowering individuals through informed decision-making. By providing accurate predictions regarding the likelihood of making a car purchase, the project facilitates a more comprehensive and educated approach to a significant financial decision. This democratization of data-driven insights ensures that a broader audience, regardless of technical expertise, can benefit from personalized recommendations, ultimately enhancing overall customer experiences in the automotive purchasing journey.

### **Business Impact:**

From a business perspective, the project promises to revolutionize marketing strategies and optimize resource allocation. The accurate predictions generated by the model enable businesses to concentrate marketing efforts on customers with a higher probability of making a car purchase. This targeted approach not only increases marketing efficiency but also contributes to cost reduction. The optimization of marketing resources directly influences Return on Investment (ROI), leading to a more efficient allocation of budget and improved financial outcomes for the business.

Moreover, the project introduces a customization feature that allows businesses to tailor the predictive model for specific market segments or product lines. This customization enhances the relevance of predictions and empowers businesses to implement highly targeted marketing



campaigns. The result is an increased likelihood of sales and a competitive advantage in the market.

## **Milestone 2: Data Collection and Visualizing and analyzing the data**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible.

### **Activity 1: Download the dataset**

There's a wealth of open-source platforms to gather data from, like kaggle.com and the UCI repository. For our project, we opted for the car\_data.csv dataset, sourced from kaggle.com. You can grab the dataset by following this link: [Car Purchase Decision - EDA and Decision Tree | Kaggle](#)

Now that we've got our hands on the dataset, let's delve into it. We'll employ visualization techniques and analytical methods to gain a comprehensive understanding of the data.

### **Activity 2: Importing the libraries**

Import the necessary libraries as shown in the image.

```
[1] # import dependencies
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# standarise the data
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

## Activity 3: Read the Dataset

Datasets come in various formats like .csv, Excel files, .txt, .json, and more. Pandas, our trusty tool, comes to the rescue for reading these datasets.

To accomplish this, we utilize the `read_csv()` function within pandas. Simply provide the directory of the CSV file as a parameter.

```
[ ] # Data collection and analysis
car_ds = pd.read_csv('/content/car_data.csv')

#printing the first 5 of dataset
car_ds.head()
```

	User ID	Gender	Age	AnnualSalary	Purchased
0	385	Male	35	20000	0
1	681	Male	40	43500	0
2	353	Male	49	74000	0
3	895	Male	40	107500	1
4	661	Male	25	79000	0

## Activity 4: Univariate analysis

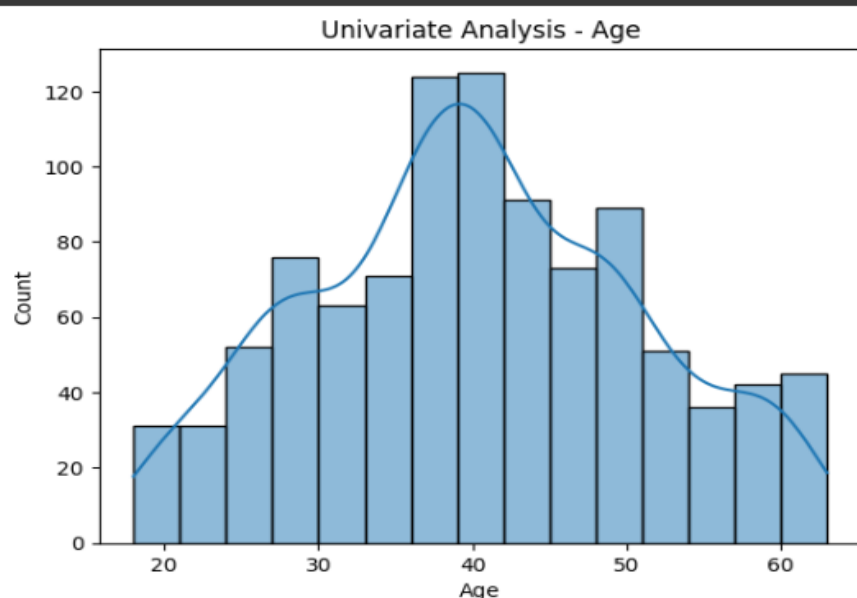
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Histplot and countplot.

Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

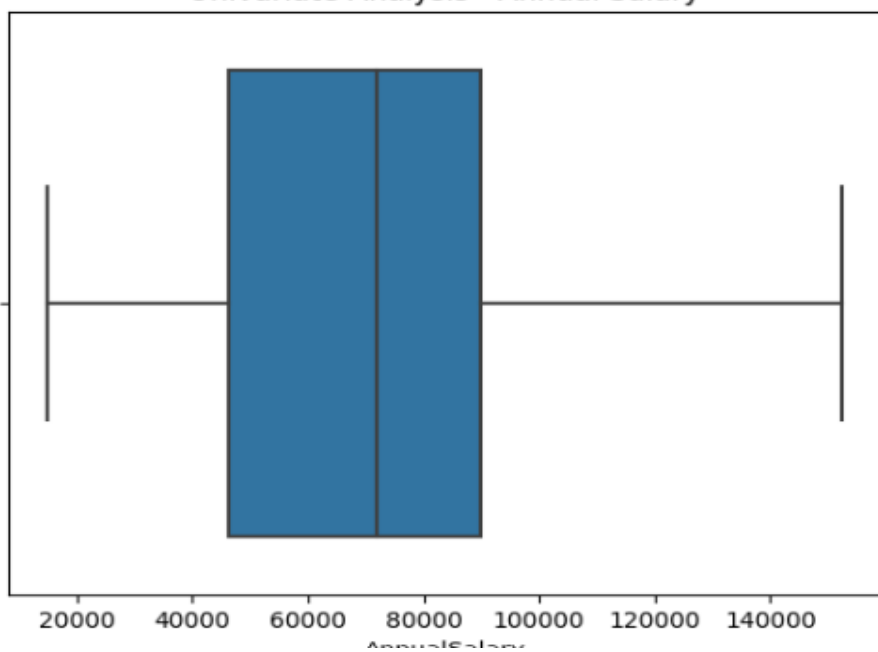
```
[ ] # Univariate analysis for 'Age'
sns.histplot(car_ds['Age'], kde=True)
plt.title('Univariate Analysis - Age')
plt.show()

# Univariate analysis for 'AnnualSalary'
sns.boxplot(x=car_ds['AnnualSalary'])
plt.title('Univariate Analysis - Annual Salary')
plt.show()

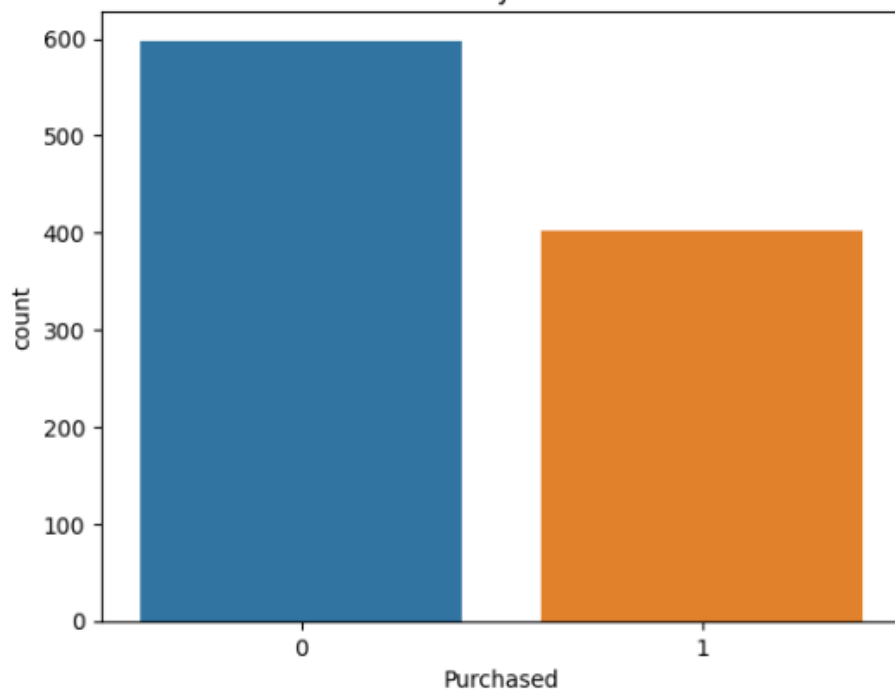
# Univariate analysis for 'Purchased' (assuming it's a categorical variable)
sns.countplot(x=car_ds['Purchased'])
plt.title('Univariate Analysis - Purchased')
plt.show()
```



Univariate Analysis - Annual Salary



Univariate Analysis - Purchased

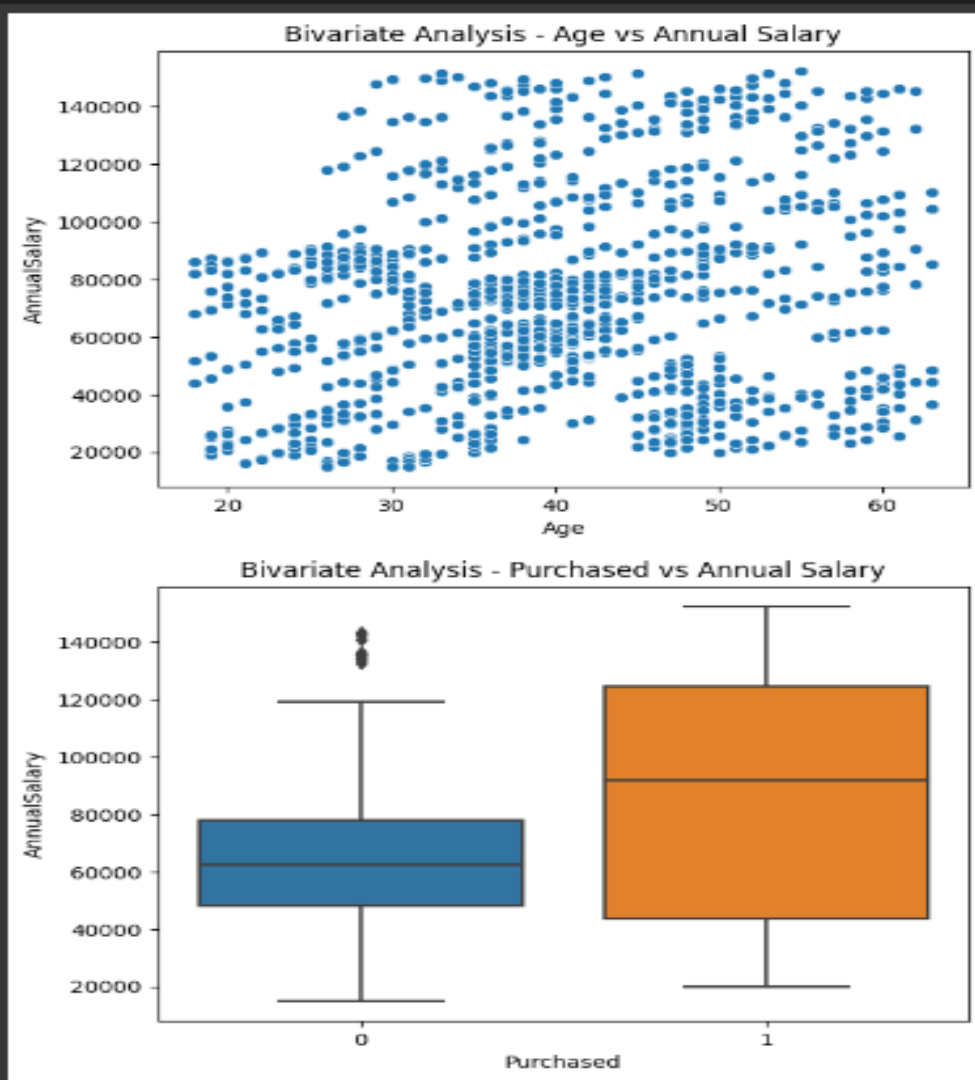


## Activity 5: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing the relationship between 'Age' and 'AnnualSalary' variables using scatterplot.

```
# Bivariate analysis for 'Age' and 'AnnualSalary'
sns.scatterplot(x='Age', y='AnnualSalary', data=car_ds)
plt.title('Bivariate Analysis - Age vs Annual Salary')
plt.show()

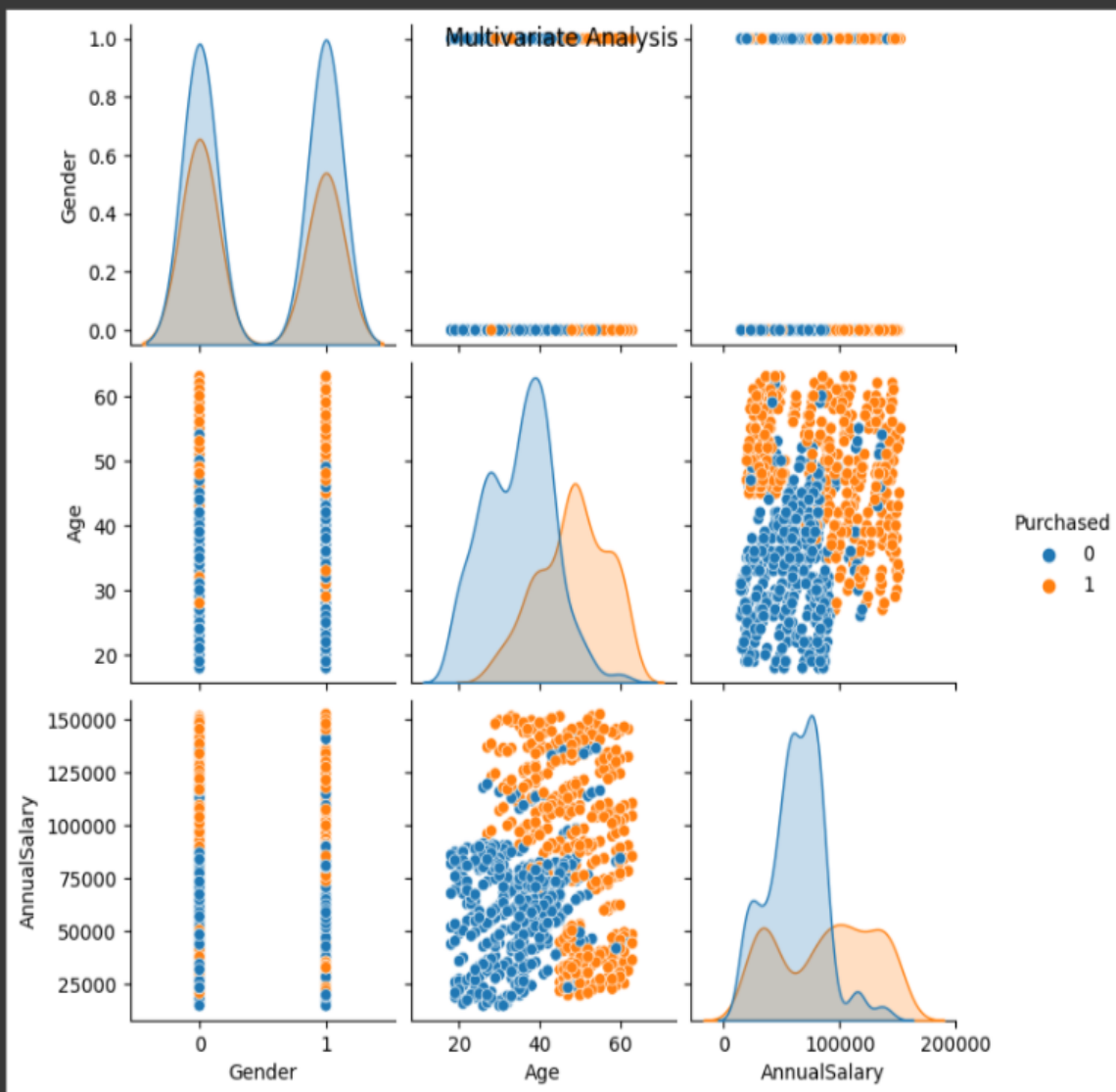
# Bivariate analysis for 'Purchased' and 'AnnualSalary'
sns.boxplot(x='Purchased', y='AnnualSalary', data=car_ds)
plt.title('Bivariate Analysis - Purchased vs Annual Salary')
plt.show()
```



## Activity 6: Multivariate analysis

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

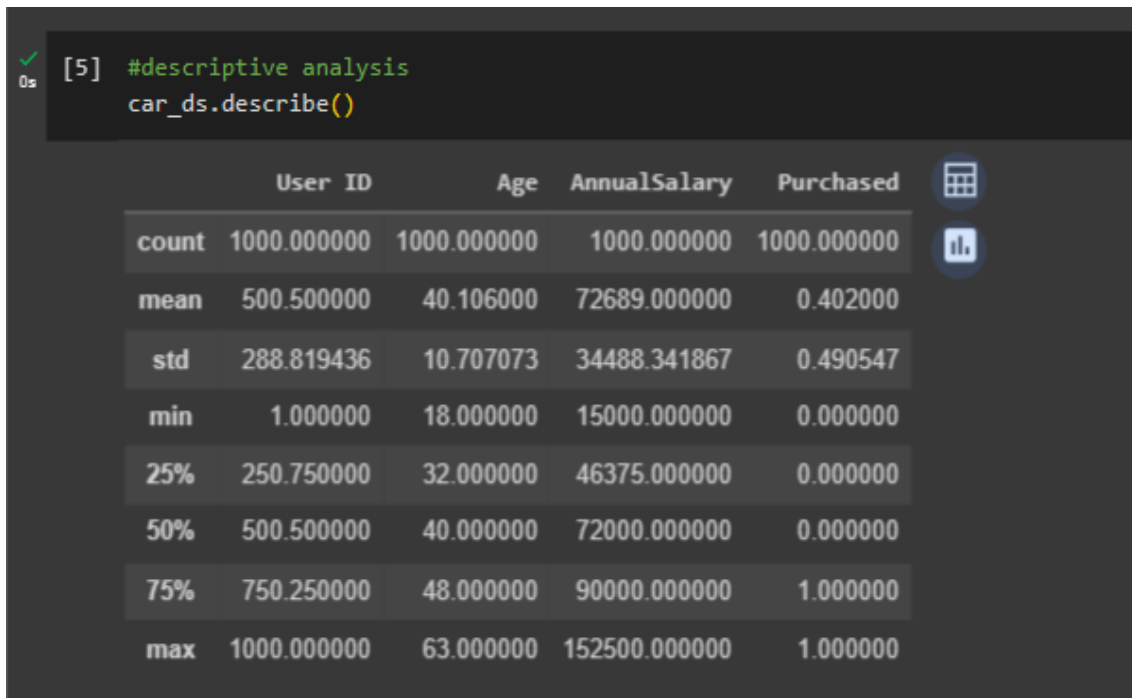
```
# Multivariate analysis using pairplot for 'Age', 'AnnualSalary', and 'Purchased'  
sns.pairplot(car_ds, hue='Purchased')  
plt.suptitle('Multivariate Analysis')  
plt.show()
```



## Activity 7: Descriptive analysis

Descriptive analysis involves delving into the fundamental aspects of data through statistical processes. In the realm of pandas, the describe function proves invaluable.

This function unveils insights into categorical features by revealing their unique, top, and frequently occurring values. Simultaneously, it unveils key statistics such as mean, standard deviation, minimum, maximum, and percentiles for continuous features.



```
[5] #descriptive analysis
car_ds.describe()
```

	User ID	Age	AnnualSalary	Purchased
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	40.106000	72689.000000	0.402000
std	288.819436	10.707073	34488.341867	0.490547
min	1.000000	18.000000	15000.000000	0.000000
25%	250.750000	32.000000	46375.000000	0.000000
50%	500.500000	40.000000	72000.000000	0.000000
75%	750.250000	48.000000	90000.000000	1.000000
max	1000.000000	63.000000	152500.000000	1.000000

## Milestone 3: Data Pre-processing

The download data set is not suitable for training the machine learning model as it might have so much of 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 categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

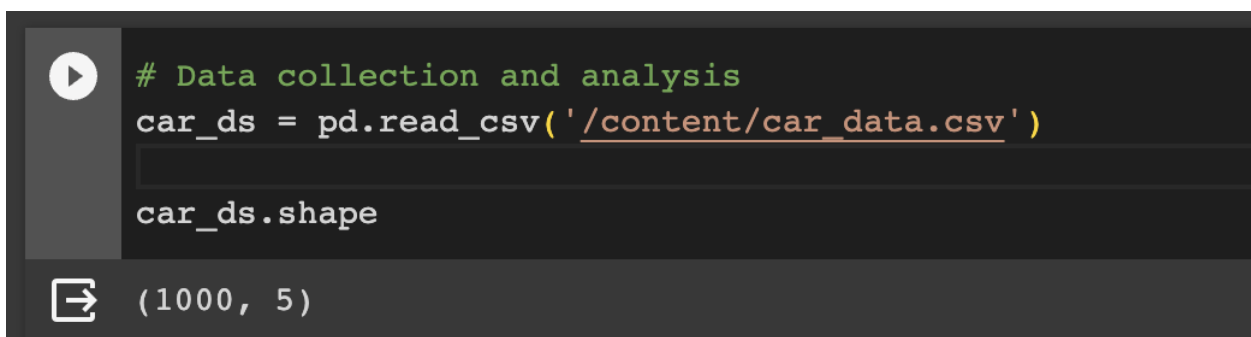
Note: These are the general steps of pre-processing the data before using it for machine learning.

Depending on the condition of your dataset, you may or may not have to go through all these steps.

### Activity 1: Checking for null values

● Let's find the shape of our dataset first, To find the shape of our data, `df.shape` method is used.

To find the data type, `df.info()` function is used.



```
# Data collection and analysis
car_ds = pd.read_csv('/content/car_data.csv')

car_ds.shape
```

(1000, 5)



```
▶ # Data collection and analysis
car_ds = pd.read_csv('/content/car_data.csv')

car_ds.info()
```

```
➞ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   User ID         1000 non-null   int64
1   Gender          1000 non-null   object
2   Age             1000 non-null   int64
3   AnnualSalary    1000 non-null   int64
4   Purchased       1000 non-null   int64
dtypes: int64(4), object(1)
memory usage: 39.2+ KB
```

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



```
# Data collection and analysis
car_ds = pd.read_csv('/content/car_data.csv')

car_ds.isnull().any()
```



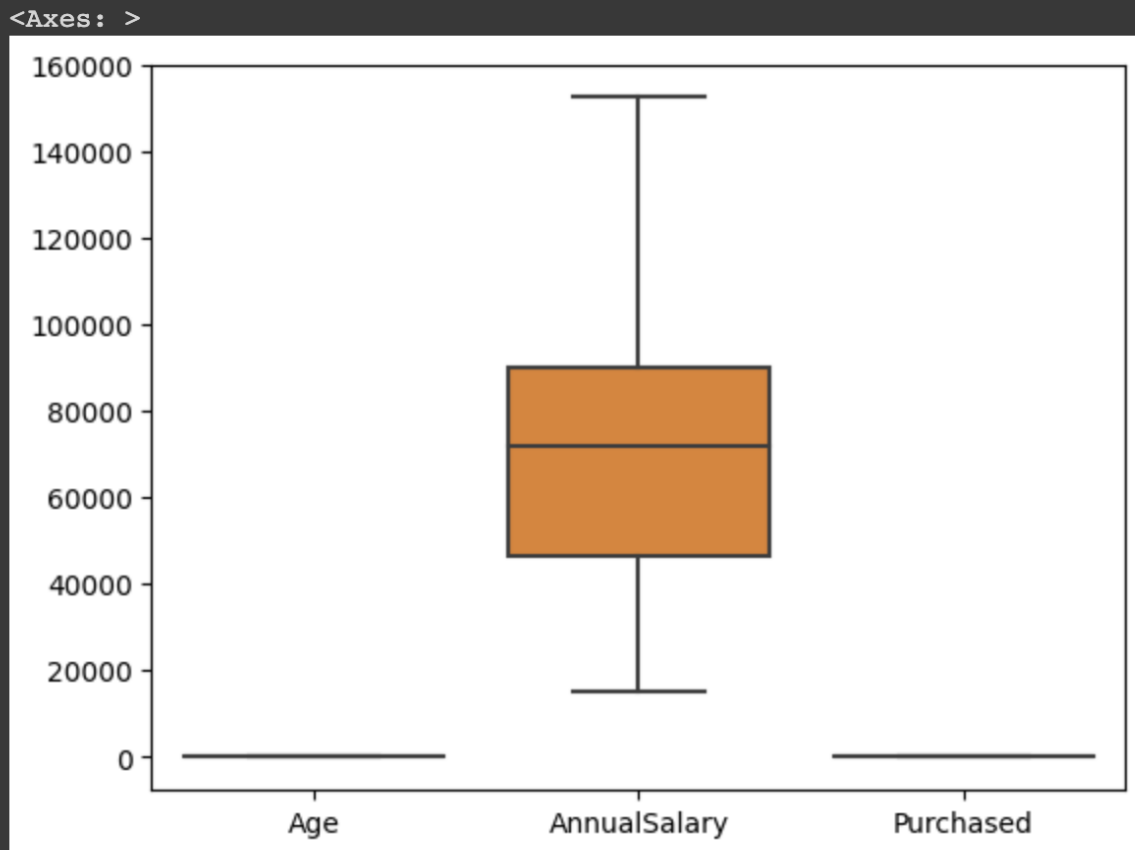
```
User ID      False
Gender       False
Age          False
AnnualSalary False
Purchased    False
dtype: bool
```

## Activity 2: Handling outliers

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

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

```
[11] sns.boxplot(car_ds)
```



### Activity 3: 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

```
# separate the data and label ( separating outcome)  
X = car_ds.drop(columns = 'Purchased',axis=1)  
Y = car_ds['Purchased']
```

```
# Data Standardization ( Standardizing the data to a particular range)  
scaler = StandardScaler()  
scaler.fit(X)
```

```
# Train - Test Split  
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2, stratify = Y,random_state =2 )  
print(X_train.shape,X_test.shape,X.shape)
```

Here x and y variables are created

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`.

## Milestone 4: Model Building

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 classification algorithms. The best model is saved based on its performance.

### Activity 1: Decision Tree Classifier

Decision Tree Classifier algorithm is initialized 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.

```
# Train - Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.2, stratify = Y,random_state =2 )
print(X_train.shape,X_test.shape,X.shape)
from sklearn.tree import DecisionTreeClassifier
DT_model = DecisionTreeClassifier(random_state=192)
DT_model.fit(X_train, Y_train)

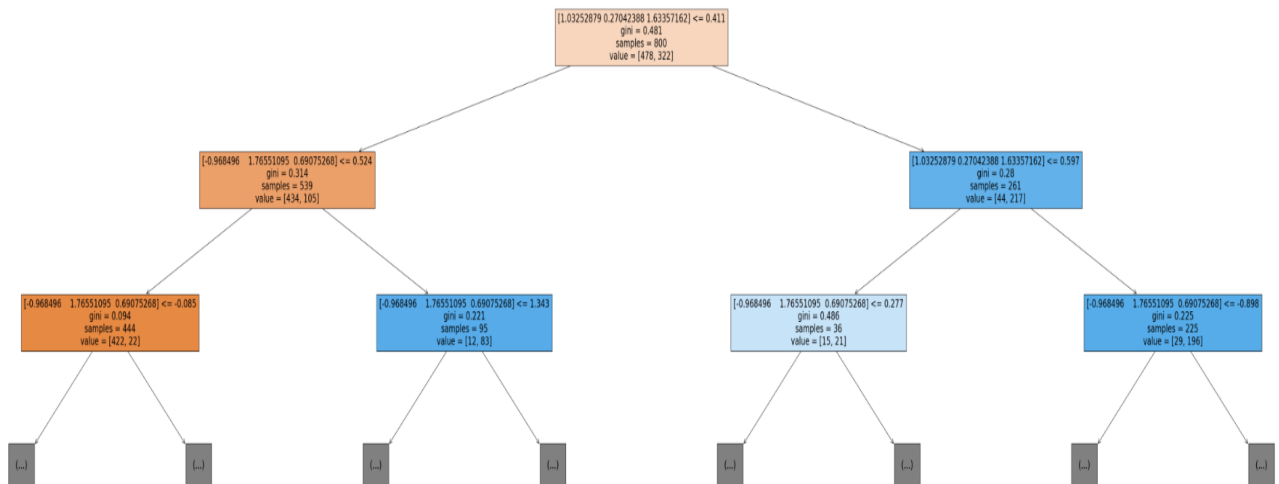
(800, 3) (200, 3) (1000, 3)
▼ DecisionTreeClassifier
DecisionTreeClassifier(random_state=192)

# Training the model
Classifier = svm.SVC(kernel = 'linear')
```

```

from sklearn.tree import plot_tree, export_text
import matplotlib.pyplot as plt
# Convert the DataFrame Index to a list of strings
feature_names = X_train
plt.figure(figsize=(80, 20))
plot_tree(DT_model, feature_names=feature_names, max_depth=2, filled=True)
plt.show()

```



## Activity 2: SVC Classifier model

SVC Classifier algorithm is initialized and training data is passed to the model with `.fit()` function.

Test data is predicted with `.predict()` function and saved in new variable.

```

# Training the model
Classifier = svm.SVC(kernel = 'linear')

```

```

# Fitting training data to classifier
Classifier.fit(X_train, Y_train)

```

▼ SVC

```
SVC(kernel='linear')
```

### Activity 3: Evaluating performance of the model

```
# Checking the accuracy
X_train_prediction = Classifier.predict(X_train)
Training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print(Training_data_accuracy)
```

```
0.82875
```

```
# Checking the accuracy of test data
X_test_prediction = Classifier.predict(X_test)
Testing_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print(Testing_data_accuracy)
```

```
0.815
```

```
# classification report
# Assuming you have X_test and Y_test
predictions = Classifier.predict(X_test)

# Generate the classification report
report = classification_report(Y_test, predictions)

# Print or use the report as needed
print(report)
```

	precision	recall	f1-score	support
0	0.83	0.87	0.85	120
1	0.79	0.74	0.76	80
accuracy			0.81	200
macro avg	0.81	0.80	0.81	200
weighted avg	0.81	0.81	0.81	200

### Activity 4: Predict the chance of purchase according to given factors

Our model is performing well. So, we are saving the model by `pickle.dump()`.

```

input = (12, 'Male', 23, 1000)
input_np = np.asarray(input)
male_mask = input_np == 'Male'
female_mask = input_np == 'Female'
input_np[male_mask] = '1'
input_np[female_mask] = '0'

input_np = input_np[1:]

print(input_np)
# reshape the array as we predict for 1 instance
input_np_reshape = input_np.reshape(1, -1)

print(input_np)

# standardize the data
std_input = scaler.transform(input_np_reshape)

# predicting the output
prediction = Classifier.predict(std_input)
print(prediction)

if(prediction == 1):
    print('The person has purchased a car')
else:
    print('The person has not purchased a car')

['1' '23' '1000']
['1' '23' '1000']
[0]
The person has not purchased a car
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have a dtype attribute
warnings.warn(

```

```

import pickle

filename = 'trained_model.sav'
pickle.dump(Classifier, open(filename, 'wb'))

#loading the model
loaded_model = pickle.load(open('trained_model.sav', 'rb'))

```

## Milestone 5: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where he has to enter the input 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

- Building HTML Pages
- Building server-side script

### Activity1: Building Html Pages:

For this project we create one HTML file namely

- index.html
- and save them in Templates folder.

```
Templates > index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Car Purchase Prediction</title>
7      <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='index.css') }}">
8      <link rel="stylesheet" type="text/css" href="https://cdn.jsdelivr.net/npm/font-awesome@5.15.1/css/all.min.css">
9
10 </head>
11 <body>
12     <nav>
13         <div class="Heading">
14             <div class="brand"><h1>Car Purchase Prediction </h1></div>
15             <ul class="menu">
16                 <span id="Haboutus"><a href="#about">About Us</a></span>
17             </ul>
18         </div>
19
20     </nav>
```



```

20 <form action="{{url_for('predict')}}", method="post" id="calculate">
21
22     <div class="inputBox">
23         <input type ='number' required="required" name="userid">
24         <span>User ID</span>
25     </div>
26     <div id="Gender">
27         <span>GENDER:</span>
28         <span id="maleG">
29             <input type="radio" name = "gender" value="Male" required="required"> Male
30         </span>
31         <span id="femaleG">
32             <input type="radio" name = "gender" value="Female" required="required" > Female
33         </span>
34     </div>
35 <div class="inputBox">
36     <input type ='number' required="required" name="age">
37     <span>Age</span>
38 </div>
39 <div class="inputBox">
40     <input type ='number' required="required" name="salary">
41     <span>Annual Salary</span>
42 </div>
43 <div>
44     <button id="predict-button" class="predictBox">Predict</button>
45 </div>
46 <div class="predictionText">{{ prediction_text }}</div>
47 </form>
48 <br><br><br>
49 <section id="about"> </section>

```

```

50 <footer class="footer">
51     <div class="container">
52         <div class="row">
53             <div class="footer-col" >
54                 <h4>About the Project</h4>
55                 <ul>
56                     <li><div class="typing"><h5>Welcome to our Car Purchase Prediction project! Here,
57                         we have created an intelligent system that assists users
58                         in predicting their car affordability based on key details
59                         like age, gender, and annual salary. Powered by machine learning,
60                         our application uses a trained model to provide personalized predictions.
61                         Explore the innovative work of our team members, and feel free to check
62                         out the project on GitHub for more details and contributions.</h5></div></li>
63                 </ul>
64             </div>
65             <div class="footer-col" id="team">
66                 <h4>Team members</h4>
67                 <ul>
68                     <li><a href="#">Kelvin J Anil</a></li>
69                     <li><a href="#">Sutam Chandra</a></li>
70                     <li><a href="#">Rishi raj Upadhyaya</a></li>
71                     <li><a href="#">Amish Ranjan</a></li>
72                 </ul>
73             </div>
74             <div class="footer-col" id="team1">
75                 <h4>See our Project on GitHub</h4>
76                 <div class="social-links">
77                     <a href="https://github.com/smartinternz02/SI-GuidedProject-600946-1697539328"><i class="fab fa-github"></i></a>
78                 </div>
79             </div>
80         </div>
81     </div>
82 </div>
83 </div>
84 </footer>

```

```

87 <script>
88     // Script for smooth scrolling to the "About Us" section
89     document.querySelectorAll('a[href^="#"]').forEach(anchor => {
90         anchor.addEventListener('click', function (e) {
91             e.preventDefault();
92             document.querySelector(this.getAttribute('href')).scrollIntoView({
93                 behavior: 'smooth'
94             });
95         });
96     });
97
98 </script>
99 </body>
100 </html>

```

## Activity 2: Build Python code:

Import the libraries

```

import pickle
from typing import Any

import numpy as np
from flask import Flask, request, render_template, jsonify
from numpy import ndarray, dtype

```

Load the saved model. Importing 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.

```
7      app = Flask(__name__)
8
9      # Load the pickled model
10     model = pickle.load(open("main.pkl", "rb"))
11
```

Render HTML page:

```
14     @app.route("/")
15     def Home():
16         return render_template("index.html")
17
```

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

In the above example, ‘/’ URL is bound with home.html function.

Hence, when the home page of the web server is opened in 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:

```

19 @app.route(rule: "/predict", methods=["POST"])
20 def predict():
21     form_values = request.form.to_dict()
22
23     # Extract values from the dictionary and convert them to the appropriate data types
24     user_id = int(form_values.get("userid", 0))
25
26     # Map 'male' to 1 and 'female' to 0 for gender
27     gender = 1 if form_values.get("gender", "").lower() == 'male' else 0
28
29     age = int(form_values.get("age", 0))
30     salary = int(form_values.get("salary", 0))
31
32     # # Remove one feature, e.g., user_id
33     # features = [gender, age, salary]
34
35     # Create a list of features
36     features = [user_id, gender, age, salary]
37
38     features_array: ndarray[Any, dtype[Any]] = np.array(features).reshape(1, -1)
39
40     prediction = model.predict(features_array)
41     result = "purchase a car" if prediction == 1 else "not purchase a car"
42
43     # Include the features in the prediction_text
44     features_str = ", ".join(map(str, features_array))
45     prediction_text = f"You should {result}. Features: {features_str}"
46
47     return render_template(template_name_or_list="index.html", prediction_text=prediction_text)


```

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 carprediction.html page earlier.

Main Function:

```

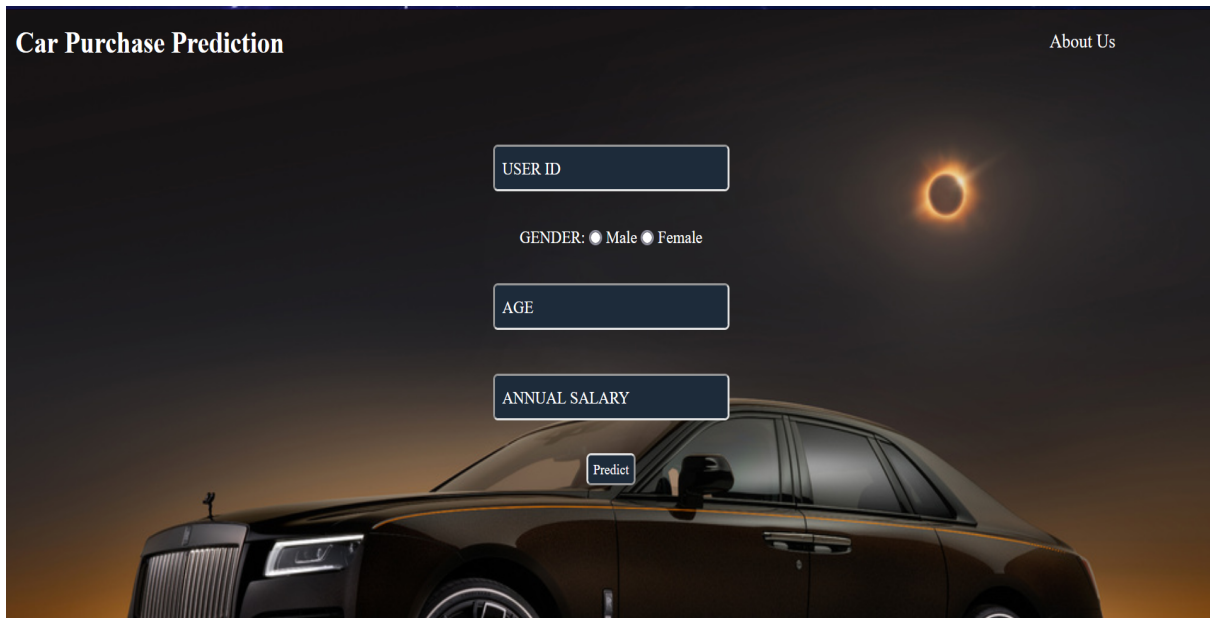
49
50  if __name__ == "__main__":
51     app.run(debug=True)

```

## Activity 3: Run the application

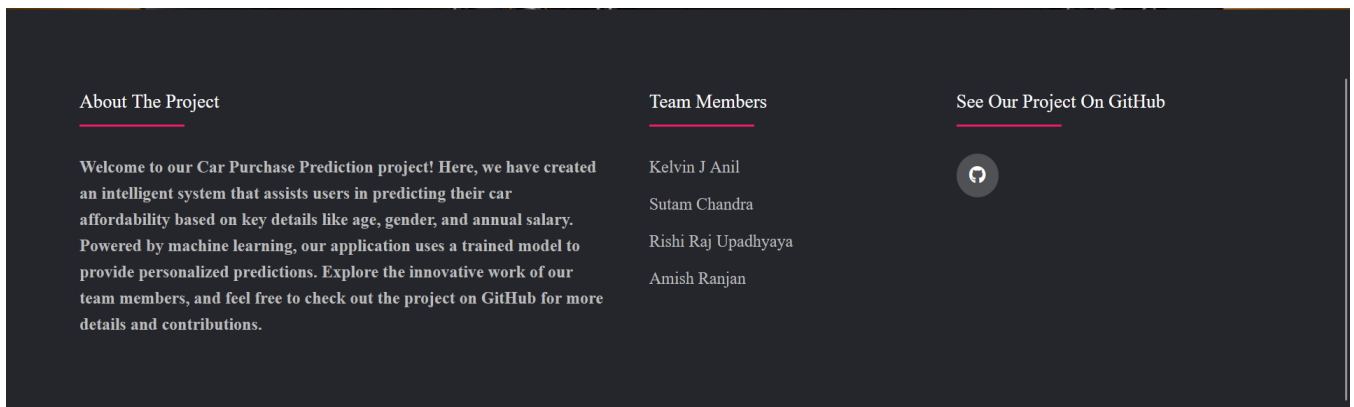
Open Visual studio code and Import all the project folders.

When you run the app.py file and click on the server url in terminal, you will redirected to our index page. And the website will looks like:



The screenshot shows a web application titled "Car Purchase Prediction" in the top left corner and "About Us" in the top right corner. The background features a dark, high-quality image of a Rolls-Royce Phantom with a solar eclipse visible in the sky. The form contains four input fields: "USER ID", "GENDER:" with radio buttons for "Male" and "Female", "AGE", and "ANNUAL SALARY". A "Predict" button is positioned below the "ANNUAL SALARY" field.

If you click on About Us option, it will take you to the footer part of our website and it looks like:



The footer section is divided into three columns. The first column, titled "About The Project", contains a welcome message and information about the machine learning model. The second column, titled "Team Members", lists the names of the four team members. The third column, titled "See Our Project On GitHub", features a GitHub logo. All titles are underlined.

About The Project	Team Members	See Our Project On GitHub
Welcome to our Car Purchase Prediction project! Here, we have created an intelligent system that assists users in predicting their car affordability based on key details like age, gender, and annual salary. Powered by machine learning, our application uses a trained model to provide personalized predictions. Explore the innovative work of our team members, and feel free to check out the project on GitHub for more details and contributions.	Kelvin J Anil Sutam Chandra Rishi Raj Upadhyaya Amish Ranjan	

Then User will enter his/her ID, Gender, Age and Annual Salary after that click on the Predict button, And our output looks like this:



The image shows a web application interface for "Car Purchase Prediction". The background features a dark, high-quality image of a Rolls-Royce Phantom with a glowing orange ring in the sky. The interface includes a header with the title "Car Purchase Prediction" on the left and a link "About Us" on the right. The form contains four input fields: "USER ID" with the value "121", "GENDER" with radio buttons for "Male" (selected) and "Female", "AGE" with the value "23", and "ANNUAL SALARY" with the value "10000". A "Predict" button is positioned below the input fields.

Car Purchase Prediction

About Us

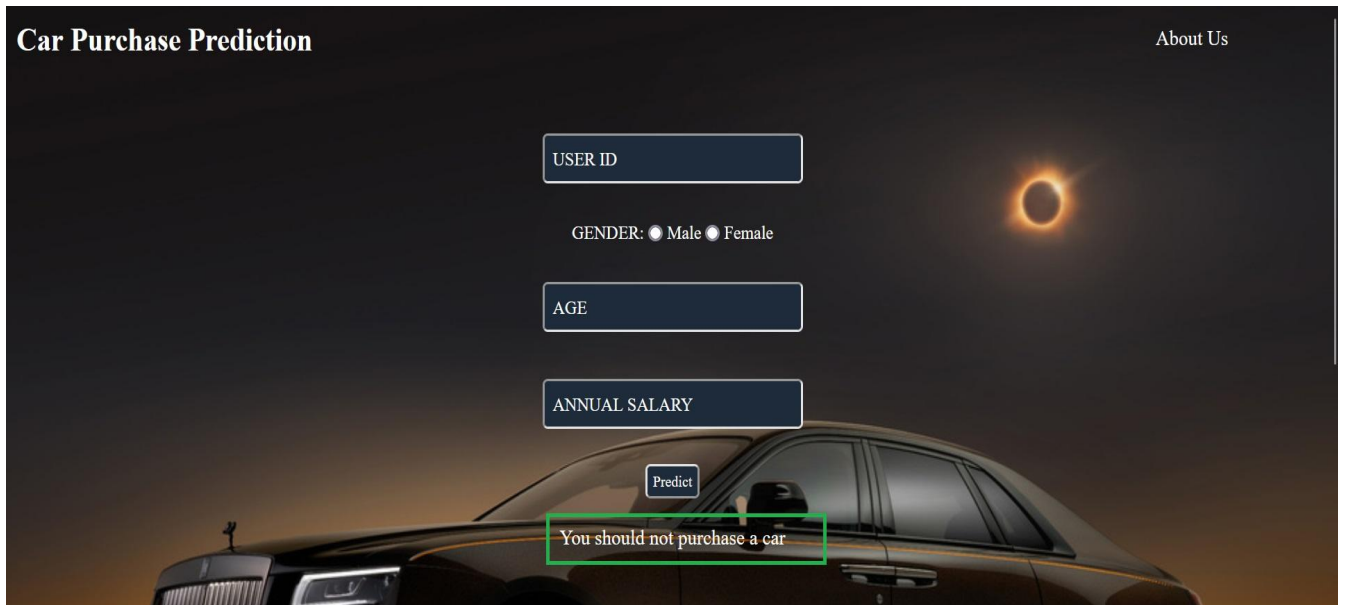
USER ID  
121

GENDER: ☒ Male ☐ Female

AGE  
23

ANNUAL SALARY  
10000

Predict



This image shows the same web application interface as the previous one, but with the "Predict" button clicked. The input fields now only contain the labels "USER ID", "AGE", and "ANNUAL SALARY". The "GENDER" section remains with the "Male" radio button selected. A green-bordered box at the bottom of the form displays the prediction result: "You should not purchase a car".

Car Purchase Prediction

About Us

USER ID

GENDER: ☒ Male ☐ Female

AGE

ANNUAL SALARY

Predict

You should not purchase a car

## **Conclusion:**

In conclusion, the developed customer car purchase prediction model proves its effectiveness in aiding decision-making within the automotive industry. Leveraging customer age and salary as predictive features, the model offers valuable insights into purchase behaviors.

Its accurate predictions empower businesses to tailor marketing strategies for targeted customer segments, optimizing resource allocation. By enabling personalized interactions, the model enhances user experience and engagement. Its integration into a user-friendly web interface simplifies access, making it an invaluable tool for both customers and dealerships. The model not only streamlines sales efforts but also drives customer satisfaction through informed choices. As a pivotal advancement in the automotive landscape, this model marks a significant step toward data-driven success.