

# Car purchase Prediction Using ML

## Project Description:

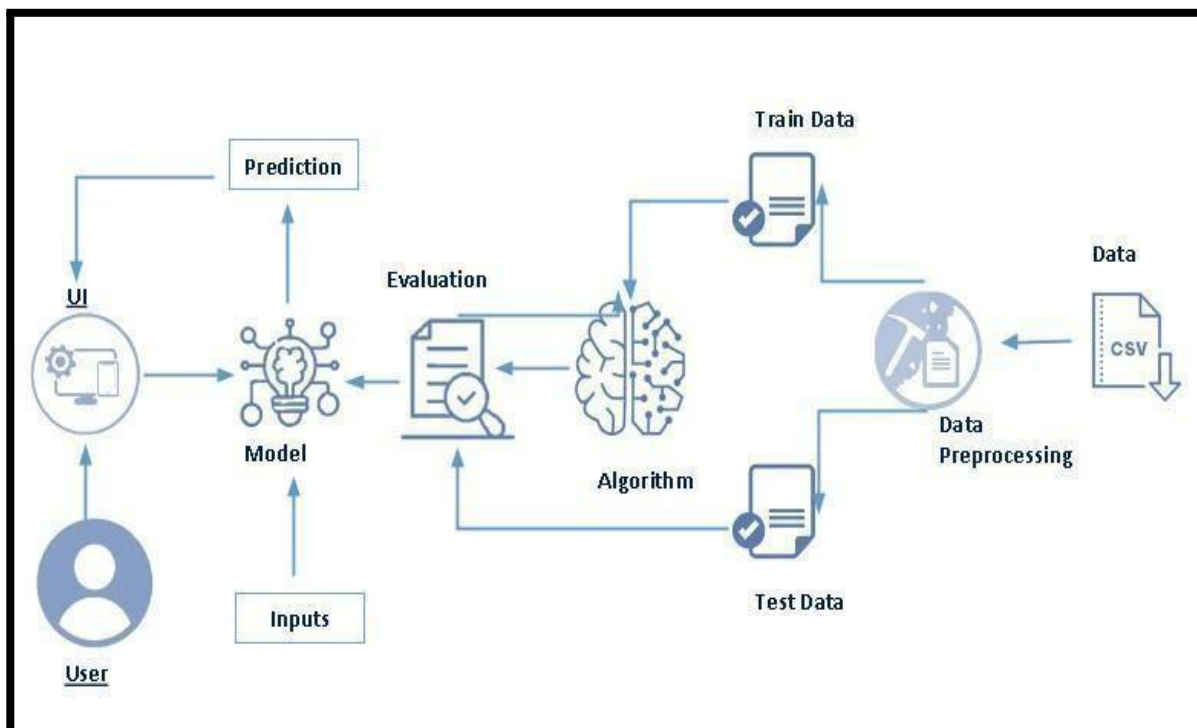
Developed an innovative ML solution to predict car purchases based on customer data. Leveraged features such as age, income, and historical purchase patterns for accurate forecasts. Achieved high predictive accuracy using advanced algorithms and thorough data preprocessing. The model assists potential buyers by estimating their likelihood to make a purchase, guiding decision-making.

Seamlessly integrated the model into a user-friendly interface, enabling easy predictions for users. This project revolutionizes the automotive industry by offering insights for tailored marketing strategies. Enhances customer experiences by facilitating informed choices and dealership targeting.

A groundbreaking application of ML driving data-powered decisions.

Through meticulous training and feature engineering, the model attains a high accuracy rate, ensuring dependable predictions. Seamlessly integrated into a user-friendly interface, users input their demographics and receive precise purchase likelihoods. This innovation revolutionizes marketing strategies by enabling targeted customer engagement and resource optimization. By harnessing data insights, the project empowers automotive businesses to tailor their approaches, enhancing overall efficiency.

## Technical Architecture:



## Pre requisites:

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

- **Visual studio code**
- **Python packages:**
  - Open Visual studio code terminal
  - 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 scipy” 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.

- **ML Concepts**

[Random forest](#)

[Decision tree](#)

[Logistic regression](#)

[SVM](#)

[Evaluation metrics](#)

- **Flask Basics :** [https://www.youtube.com/watch?v=lj4I\\_CvBnt0](https://www.youtube.com/watch?v=lj4I_CvBnt0)

## Project Objectives:

Develop a high-accuracy machine learning solution to predict car purchases based on customer data, offering a user-friendly interface. This project aims to revolutionize marketing strategies, enhance customer experiences, and empower automotive businesses to optimize their approach by harnessing data insights.

## **Project Flow:**

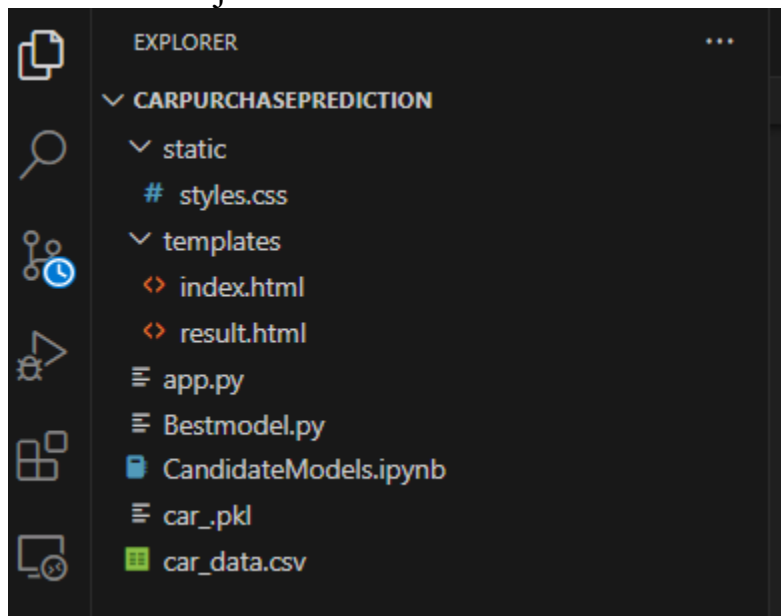
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

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

- Data collection
  - o Collect the dataset or create the dataset
- Visualizing and analyzing data
  - o Univariate analysis
  - o Bivariate analysis
  - o Multivariate analysis
  - o Descriptive analysis
- Data pre-processing
  - o Checking for null values
  - o Handling outlier
  - o Handling categorical data
  - o Splitting data into train and test
- Model building
  - o Import the model building libraries
  - o Initializing the model
  - o Training and testing the model
  - o Evaluating performance of model
  - o Save the model
- Application Building
  - o Create an HTML file
  - o 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.
- car\_.pkl is our saved model. Further we will use this model for flask integration.
- Data folder contains dataset and Templates folder contains html files.

## Milestone 1: Define Problem / Problem Understanding

### Activity 1: Specify the business problem

The business problem addressed by the car purchase prediction ML model is to enhance the efficiency of marketing and sales strategies for automotive companies. The primary challenge is to identify potential customers who are more likely to make a car purchase based on their age and income demographics. By accurately predicting customer purchase behaviors, the model assists businesses in allocating their marketing resources more effectively. This optimization leads to personalized targeting of potential customers, reducing marketing costs while increasing the chances of successful conversions. Overall, the model aims to provide a data-driven solution that enhances the decision-making process in the automotive industry and maximizes the return on marketing investments.

## Activity 2: Business requirements

1. **Data Collection and Integration:** Acquire and integrate a comprehensive dataset that includes customer demographic information, income, and historical purchase patterns, ensuring the accuracy and consistency of the data.
2. **Integration with Marketing Strategies:** Ensure seamless integration of the car purchase prediction model with the company's marketing strategies. This should empower the development of targeted marketing campaigns and dealership targeting efforts aimed at potential car buyers.
3. **Scalability:** Design the car purchase prediction model to accommodate a growing number of customer data points while maintaining high prediction accuracy and fast response times. The system should be able to handle an increasing volume of data effectively.
4. **Resource Optimization:** Support marketing resource optimization by guiding efforts and resources towards customers with a higher probability of making a car purchase. This should lead to cost reduction and an enhanced return on investment (ROI).
5. **Customization:** Provide the capability to customize the car purchase prediction model to cater to specific market segments or product lines. This flexibility enables fine-tuned predictions and allows for the development of targeted marketing and sales strategies tailored to different customer segments.

## Activity 3: Literature Survey

### Literature Review: Car Purchase Prediction with Machine Learning

#### Introduction:

The literature on car purchase prediction using machine learning (ML) explores the application of data-driven models to forecast and understand consumer behavior in the automotive industry. This review aims to provide insights into existing research, methodologies, and key findings in the domain of ML-driven car purchase prediction.

#### Historical Overview:

Early studies in car purchase prediction primarily relied on traditional statistical methods. However, with the advent of machine learning techniques, researchers began adopting predictive modeling to enhance the accuracy and efficiency of car purchase predictions. This shift marked a transformative period in the field, unlocking new possibilities for more sophisticated analyses.

**Theoretical Foundations:**

Several ML algorithms form the theoretical foundations for car purchase prediction. Algorithms such as regression models, decision trees, support vector machines, and neural networks have been applied to model complex relationships between predictor variables and the likelihood of car purchases. Understanding the strengths and limitations of these algorithms is essential for advancing predictive modeling in the automotive sector.

**Key Predictive Features:**

Studies emphasize the identification and selection of predictive features that significantly influence car purchase decisions. Demographic information, socio-economic factors, online behavior, and historical purchase data are among the key features used to train ML models. The exploration of these features contributes to a more nuanced understanding of the decision-making process.

**Environmental and Economic Influences:**

The literature delves into broader contextual factors, including environmental and economic considerations, that shape car purchase behavior. Insights into how population growth, economic indicators, and disposable income impact consumer preferences contribute to building comprehensive predictive models.

**Regional and Cultural Variations:**

Some studies focus on regional and cultural nuances in car purchase behavior. Analyzing how cultural factors influence ML predictions helps tailor models to specific markets, acknowledging the diversity in consumer preferences and decision-making processes.

**Marketing and Advertising Strategies:**

Researchers investigate the impact of marketing and advertising strategies on car purchase predictions. ML models analyze the effectiveness of digital campaigns, television advertisements, and other promotional efforts, providing valuable insights for crafting targeted marketing strategies.

**Decision-Making Duration:**

Understanding the time consumers spend in the decision-making process is crucial. ML models contribute to quantifying and analyzing the factors that contribute to the duration of the decision-making process, offering valuable information for marketers and manufacturers.

**Conclusion:**

In conclusion, the literature on car purchase prediction using machine learning reflects a dynamic and evolving field. The integration of sophisticated algorithms, consideration of diverse predictors, and exploration of contextual factors contribute to the advancement of predictive modeling in the automotive industry.

## References

[https://www.researchgate.net/publication/353954063\\_Car\\_Buying\\_Behaviour\\_in\\_India](https://www.researchgate.net/publication/353954063_Car_Buying_Behaviour_in_India)

## Activity 4: Social and Business Impact.

### **Social Impact:**

The introduction of the car purchase prediction ML model brings notable social implications. Customers benefit from a more personalized experience as their purchasing likelihood is assessed based on individual characteristics, reducing irrelevant marketing outreach. This enhances user satisfaction and trust, fostering positive brand-consumer relationships. Moreover, the model encourages responsible consumption by aligning customers with suitable car options, considering their financial circumstances. As data-driven decisions become more prevalent, it encourages an informed approach to car purchases, promoting financial prudence among consumers.

### **Business Impact:**

On the business front, the car purchase prediction ML model yields substantial impact. Marketing efforts become laser-focused, targeting individuals with a higher chance of conversion, resulting in enhanced efficiency and cost reduction. The model facilitates improved resource allocation, optimizing budget allocation for campaigns that promise the highest return on investment. Sales teams can prioritize leads, streamlining the conversion process and potentially increasing sales volume. Over time, the model's accurate predictions contribute to higher customer satisfaction rates and improved brand reputation. As data-driven strategies gain prominence, the business stays ahead in a competitive market, poised for growth and innovation.

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

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

### Activity 1: Download the dataset

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

Link:<https://www.kaggle.com/code/vishesh1412/car-purchase-decision-eda-and-decision-tree/input>.

## Activity 2: Importing the libraries

```
import numpy as np
import pandas as pd
```

```
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.preprocessing import LabelEncoder
le =LabelEncoder()
```

```
# Train test split

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(scaled_x,y,test_size = 0.2,random_state = 0)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,roc_auc_score,roc_curve
print('Testing accuracy = ',accuracy_score(y_test,pred))
```



### Activity 3: Read the Dataset

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

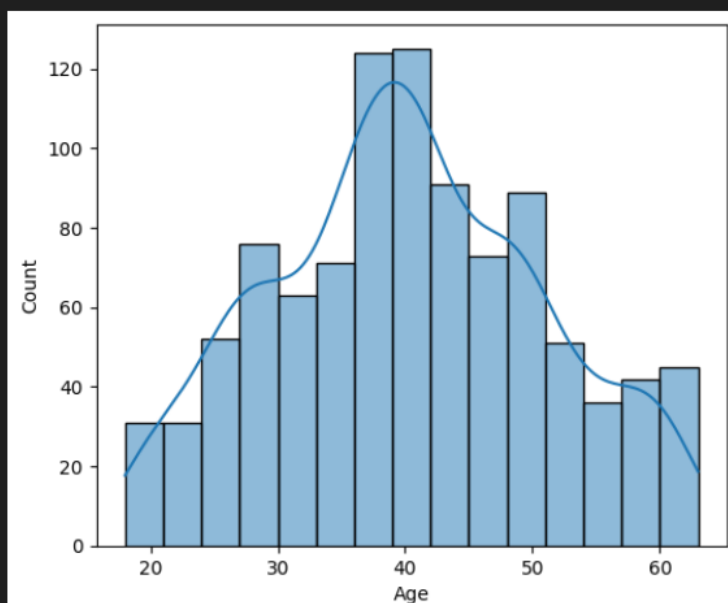
```
df = pd.read_csv('/content/car_data.csv')
df.head()
```

	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

understanding the data with single feature.

```
fig = plt.figure(figsize = (6, 5))
sns.histplot(data = df, x = "Age", kde = True)
plt.show()
```

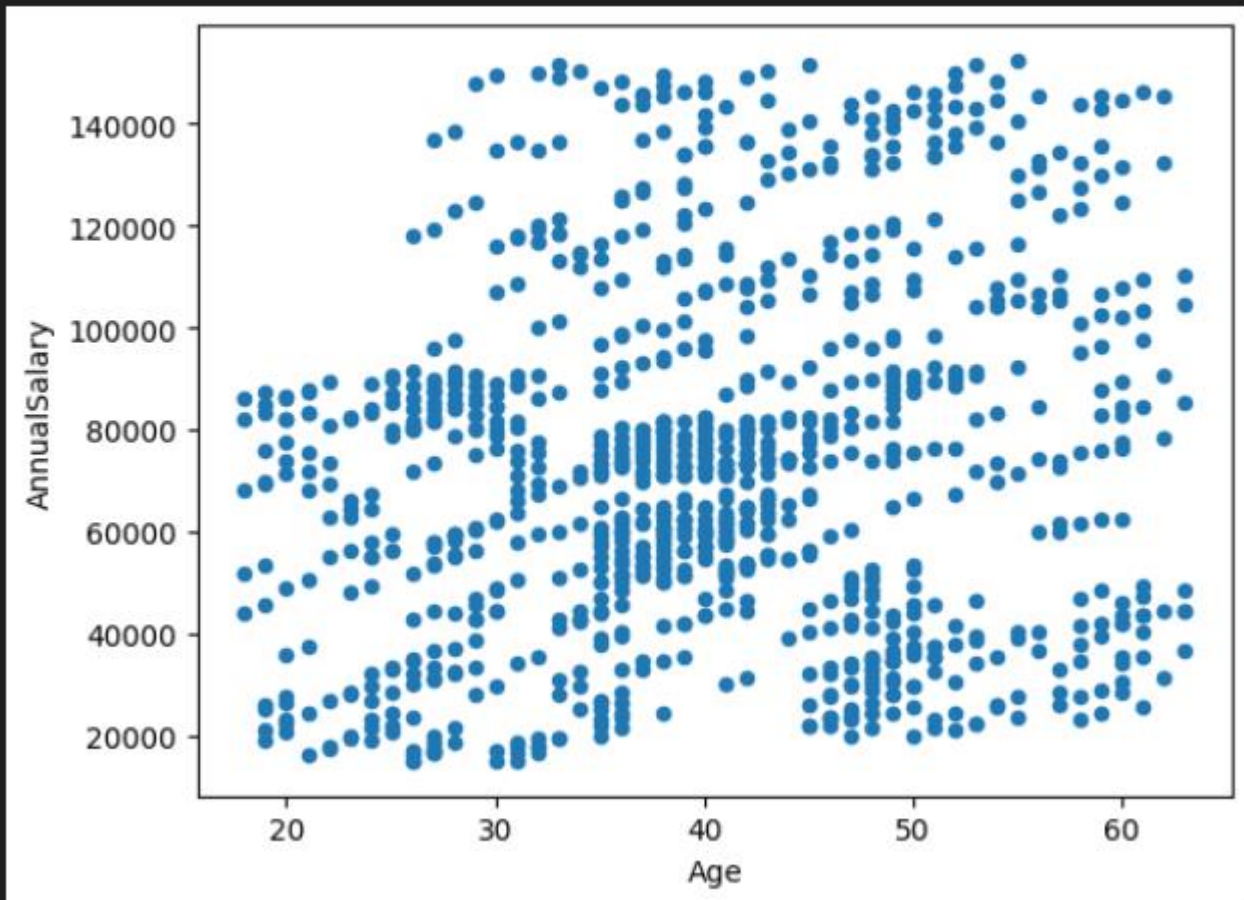


## Activity 5: Bivariate analysis

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

```
df.plot.scatter(x='Age', y='AnnualSalary')
```

```
<Axes: xlabel='Age', ylabel='AnnualSalary'>
```

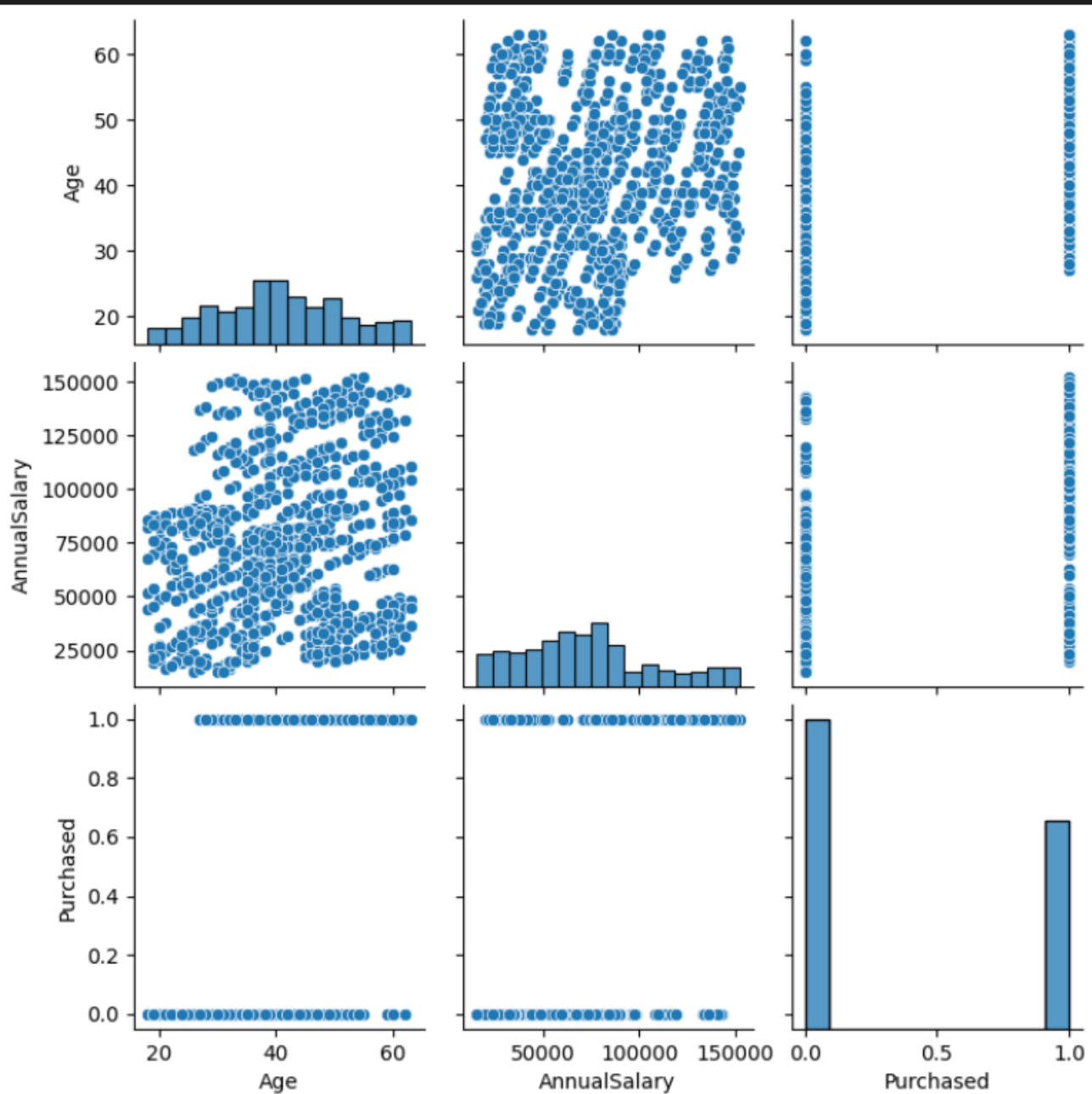


## Activity 6: Multivariate analysis

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

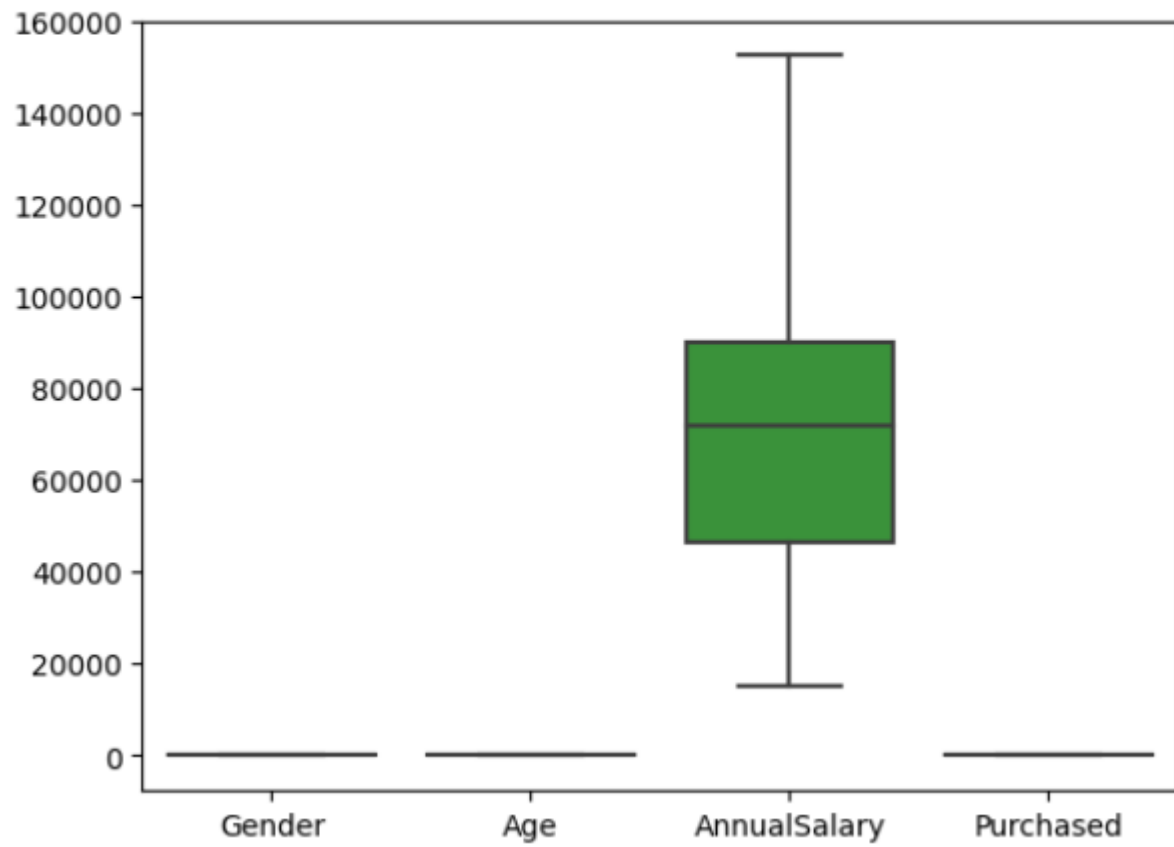
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x78fc93fcb9a0>
```



```
sns.boxplot(df)
```

<Axes: >



## Activity 7: Descriptive analysis

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.

```
df.describe()
```

	Gender	Age	AnnualSalary	Purchased
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.484000	40.106000	72689.000000	0.402000
std	0.499994	10.707073	34488.341867	0.490547
min	0.000000	18.000000	15000.000000	0.000000
25%	0.000000	32.000000	46375.000000	0.000000
50%	0.000000	40.000000	72000.000000	0.000000
75%	1.000000	48.000000	90000.000000	1.000000
max	1.000000	63.000000	152500.000000	1.000000

## Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

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

## Activity 1: Checking for null values

```
df.isnull().sum()
```

```
User ID      0  
Gender       0  
Age          0  
AnnualSalary 0  
Purchased    0  
dtype: int64
```

No null values

## Activity 2: Dropping unnecessary columns

```
df = df.drop(columns = ['User ID'],axis = 1)
```

```
df.head()
```

	Gender	Age	AnnualSalary	Purchased
0	Male	35	20000	0
1	Male	40	43500	0
2	Male	49	74000	0
3	Male	40	107500	1
4	Male	25	79000	0

### Activity 3: Handling categorical data

Here gender column is categorical . we will make it as numerical type by doing label encoding

#### Label encoding

```
from sklearn.preprocessing import LabelEncoder  
le =LabelEncoder()
```

```
df.Gender = le.fit_transform(df.Gender)
```

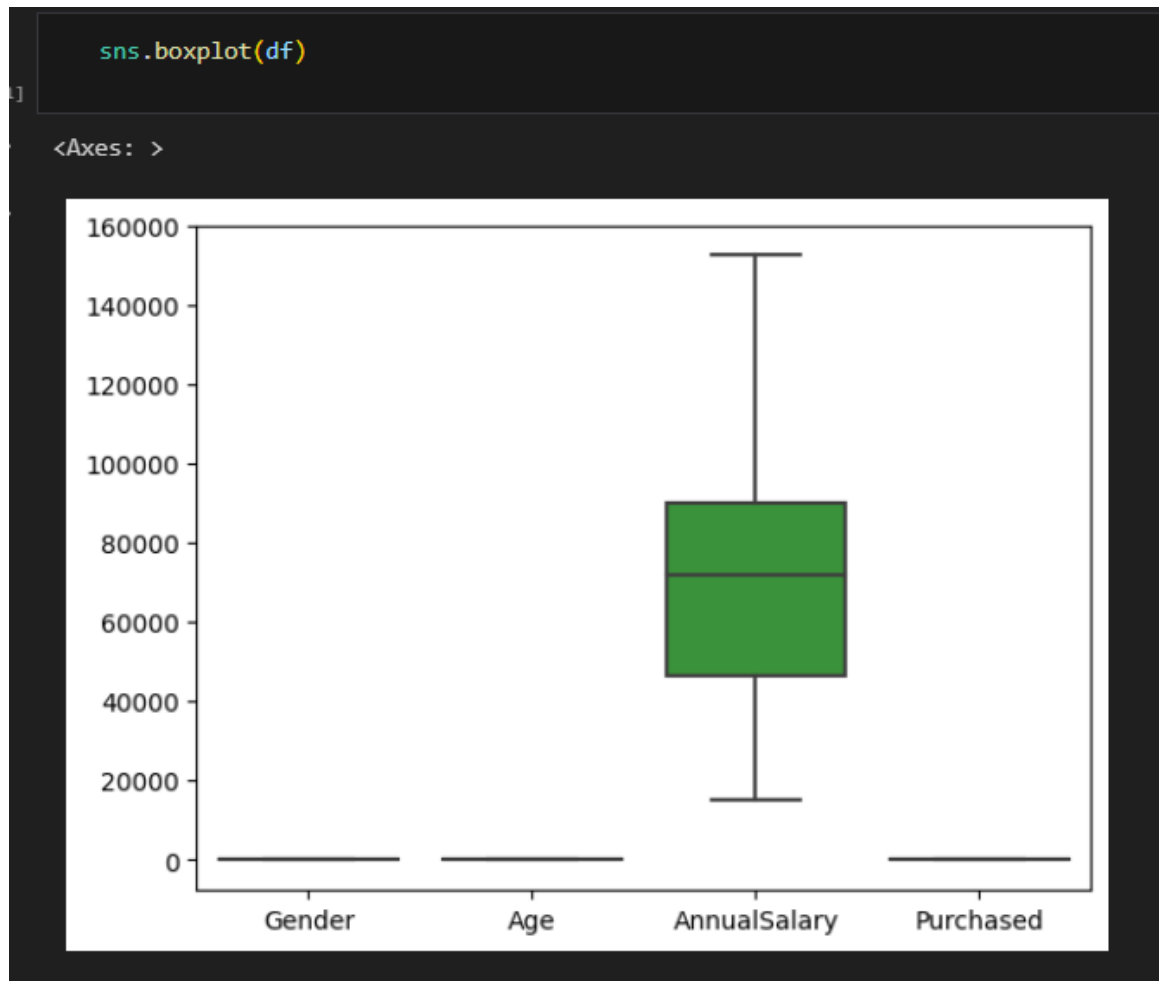
```
df.head()
```

	Gender	Age	AnnualSalary	Purchased
0	1	35	20000	0
1	1	40	43500	0
2	1	49	74000	0
3	1	40	107500	1
4	1	25	79000	0

## Activity 4: Handling outliers

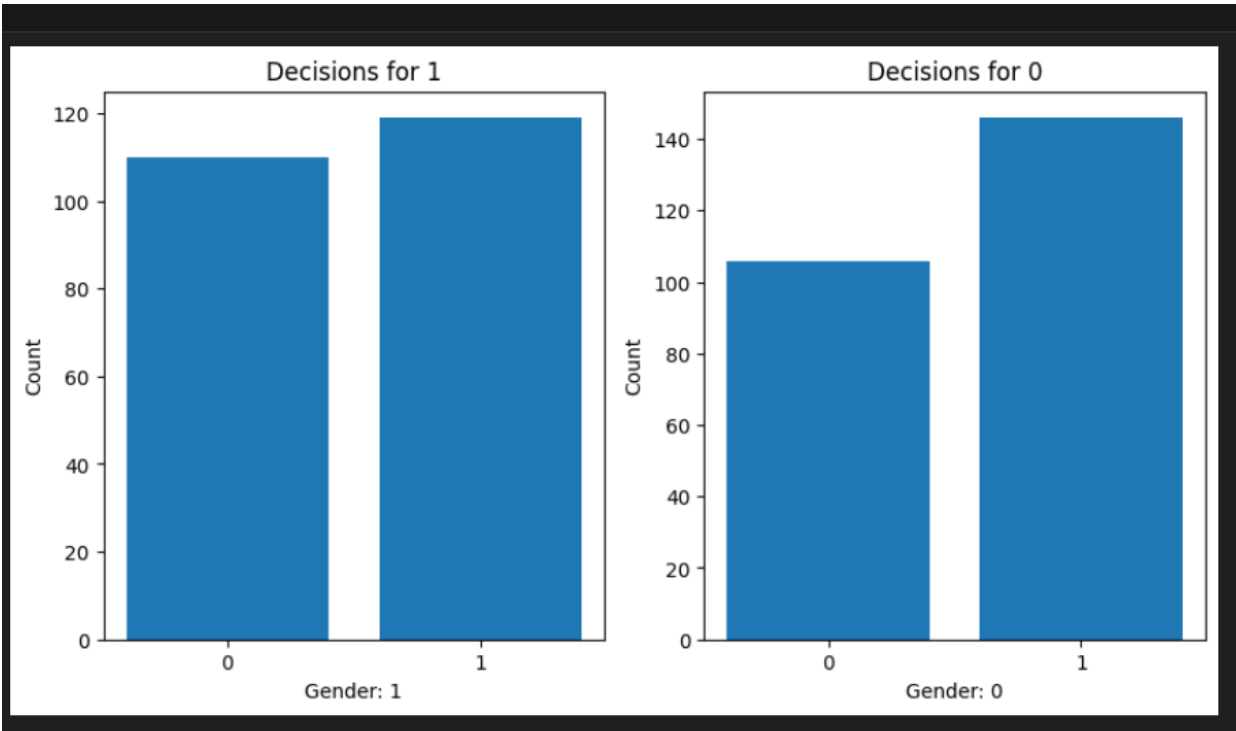
With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of all features with some mathematical formula.

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





**Activity 5: Decision made by people whose salary is greater than average salary**



1: Male , 0: Female

## Activity 6: 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`.

### Data Scaling and Splitting

```
## X and y split
```

```
X =df.drop(columns =['Purchased'],axis =1)
```

```
X.head()
```

	Gender	Age	AnnualSalary
0	1	35	20000
1	1	40	43500
2	1	49	74000
3	1	40	107500
4	1	25	79000

```
y =df.Purchased  
y.head()
```

```
0    0  
1    0  
2    0  
3    1  
4    0  
Name: Purchased, dtype: int64
```

```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()
```

```
scaled_x = pd.DataFrame(scale.fit_transform(X),columns =X.columns)
scaled_x.head()
```

	Gender	Age	AnnualSalary
0	1.0	0.377778	0.036364
1	1.0	0.488889	0.207273
2	1.0	0.688889	0.429091
3	1.0	0.488889	0.672727
4	1.0	0.155556	0.465455

Train test split

```
# Train test split
```

```
from sklearn.model_selection import
```

```
train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(scaled_x,y,test_size = 0.2,random_state = 0)
```

```
x_train.shape
```

```
(800, 3)
```

```
x_test.shape
```

```
(200, 3)
```

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

### Activity 1: Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train,y_train)
pred = model.predict(x_test)
pred
y_test
```

```
▼ LogisticRegression
LogisticRegression()
```

# Metrics

```
Regression Model Metrics:
MAE: 0.14
MSE: 0.14
RMSE: 0.37416573867739417
R2 Score: 0.4141646615754787

Classification Model Metrics:
Confusion Matrix:

[[114   7]
 [ 21  58]]
Accuracy:
0.86

Classification Report:

              precision    recall  f1-score   support

     0       0.84         0.94         0.89         121
     1       0.89         0.73         0.81          79

 accuracy          0.86         0.86         0.86         200
  macro avg       0.87         0.84         0.85         200
weighted avg       0.86         0.86         0.86         200
```

## Activity 2: Random Forest classifier



```
from sklearn.ensemble import RandomForestClassifier
```

[55]

```
from sklearn.metrics import confusion_matrix, accuracy_score
```

[56]

```
model1=RandomForestClassifier(criterion='entropy')  
model1.fit(x_train,y_train)
```

[63]

...

```
RandomForestClassifier  
RandomForestClassifier(criterion='entropy')
```

```
y_predict_1=model1.predict(x_test)  
y_predict_train=model1.predict(x_train)
```

[64]

```
y_predict_1
```

[65]

## Metric

### Regression Model Metrics:

MAE: 0.075

MSE: 0.075

RMSE: 0.27386127875258304

R2 Score: 0.6861596401297207

### Classification Model Metrics:

#### Confusion Matrix:

```
[[112  9]
 [ 8 71]]
```

Accuracy: 0.915

#### Classification Report:

	precision	recall	f1-score	support
0	0.93	0.93	0.93	121
1	0.89	0.90	0.89	79
accuracy			0.92	200
macro avg	0.91	0.91	0.91	200
weighted avg	0.92	0.92	0.92	200

### Activity 3: Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
model2=DecisionTreeClassifier(max_depth=4,splitter='best',criterion='entropy')
model2.fit(x_train,y_train)
y_predict_2=model2.predict(x_test)
y_test
y_predict_2
y_predict_train=model2.predict(x_train)
```

### Metrics

Regression Model Metrics:

MAE: 0.11802618416066693

MSE: 0.06098884234518473

RMSE: 0.2469591916596439

R2 Score: 0.7447898636042066

Classification Model Metrics :

Confusion Matrix:

```
[[116   5]
 [ 10  69]]
```

Accuracy:

0.925

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.96	0.94	121
1	0.93	0.87	0.90	79
accuracy			0.93	200
macro avg	0.93	0.92	0.92	200
weighted avg	0.93	0.93	0.92	200



## Activity 4: Kernel SVM

```
from sklearn.svm import SVC #Kernel SVM
# Create an SVM model with a linear kernel
model3 = SVC(kernel='linear')
# Fit the model on the training data
model3.fit(x_train, y_train)
y_test
# Make predictions on the test data
y_predict_3 = model3.predict(x_test)
# Make predictions on the training data (if needed)
y_predict_train = model3.predict(x_train)
```

```
from sklearn.svm import SVC #Kernel SVM
# Create an SVM model with a linear kernel
model3 = SVC(kernel='linear')
# Fit the model on the training data
model3.fit(x_train, y_train)
```

▼ SVC

SVC(kernel='linear')

# Metrics

```
Regression Model Metrics:

MAE: 0.135
MSE: 0.135
RMSE: 0.3674234614174767
R2 Score: 0.43508735223349737

Classification Model Metrics:

Confusion Matrix:
[[116  5]
 [ 22 57]]

Accuracy: 0.865

Classification Report:
              precision    recall  f1-score   support

     0       0.84      0.96      0.90      121
     1       0.92      0.72      0.81      79

 accuracy          0.86      200
  macro avg       0.88      0.84      0.85      200
 weighted avg     0.87      0.86      0.86      200
```

### Activity 5: The best Model

The best model as per the accuracy rate is The Decision Tree classifier model

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

```
import pickle

pickle.dump(model3, open('car_.pkl', 'wb'))
```

### 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 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

- Building HTML Pages
- Building serverside script

#### Activity1: Building Html Pages:

For this project create three HTML files namely

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

index.html

```
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      <link rel="stylesheet" type="text/css" href="/static/styles.css">
7      <title>Predict</title>
8  </head>
9  <body>
10     <h1>Car purchase prediction</h1>
11     <form action ="/predict" method = "post">
12
13         <label for ="Gender">Choose the Gender</label>
14         <select name = "s">
15             <option value ="cal">Male</option>
16             <option value ="flo">Female</option>
17         </select>
18         <p>Age</p>
19         <p><input type ="text" name ="rd"/></p>
20
21         <p>Salary</p>
22         <p><input type ="text" name ="as"/></p>
23
24
25
26         <p><input type ="submit" value ="submit"/></p>
27
28
29
30
31     </form>
32 </body>
33 </html>
```

## result.html

```
templates > result.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      <link rel="stylesheet" type="text/css" href="/static/styles.css">
7      <title>Prediction result</title>
8  </head>
9  <body>
10     <h1>Car purchase prediction</h1>
11     <form action ="/predict" method = "post">
12
13         <label for ="Gender">Choose the Gender</label>
14         <select name = "s">
15             <option value ="cal">Male</option>
16             <option value ="flo">Female</option>
17         </select>
18         <p>Age</p>
19         <p><input type ="text" name ="rd"/></p>
20
21         <p>Salary</p>
22         <p><input type ="text" name ="as"/></p>
23
24
25         <p><input type ="submit" value ="submit"/></p>
26     </form>
27     <div class="message-box {% if y %}success{% else %}error{% endif %}">
28         <b>{% if y %}Yes, the customer will purchase the car{% else %}No, the customer will not purchase the car{% endif %}</b>
29     </div>
30
31
32 </body>
33 </html>
```

## Activity 2: Build Python code:

### Import the libraries

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.

```
app.py
1  from flask import Flask, render_template, request
2  import pickle
3  import numpy as np
```

## Render HTML page:

```
10 @app.route('/')
11 def start():
12     return render_template('index.html')
13
```

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 predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the result.html page earlier.

```
14 @app.route('/predict', methods=['POST'])
15 def predict():
```

## Main Function:

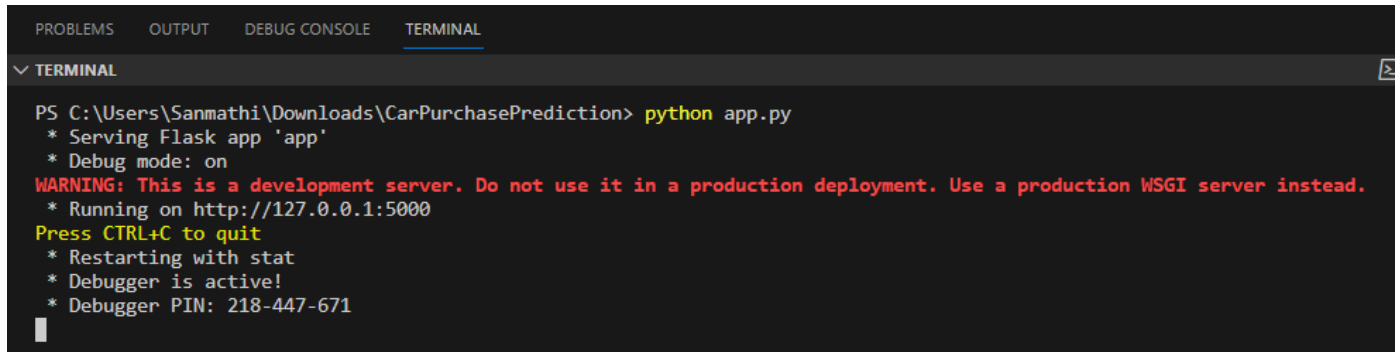
```
if __name__ == "__main__":
    app.run(debug=True)
```

## app.py

```
1  from flask import Flask, render_template, request
2  import pickle
3  import numpy as np
4
5  # Load the trained model
6  model1 = pickle.load(open('car_.pkl', 'rb'))
7
8  app = Flask(__name__)
9
10 @app.route('/')
11 def start():
12     return render_template('index.html')
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     rd = int(request.form['rd'])
17     as_val = int(request.form['as'])
18     s = request.form['s']
19
20     # Add age and salary limits
21     age_limit = 18 # Minimum age limit
22     salary_limit = 14000 # Minimum salary limit
23
24     # Convert state to binary (Male=1, Female=0)
25     if s == 'cal':
26         s = 1
27     elif s == 'flo':
28         s = 0
29
30     # Check if input values meet the specified limits
31     if rd >= age_limit and as_val >= salary_limit:
32         input_data = [[s, rd, as_val]]
33         prediction = model1.predict(input_data)
34         prediction_bool = bool(prediction[0])
35     else:
36         prediction_bool = False # If input values do not meet the limits
37
38     return render_template('result.html', y=prediction_bool)
39
40
41 if __name__ == '__main__':
42     app.run(debug=True)
```

### Activity 3: Run the application

Open Visual studio code and Import all the project folders, and do this.

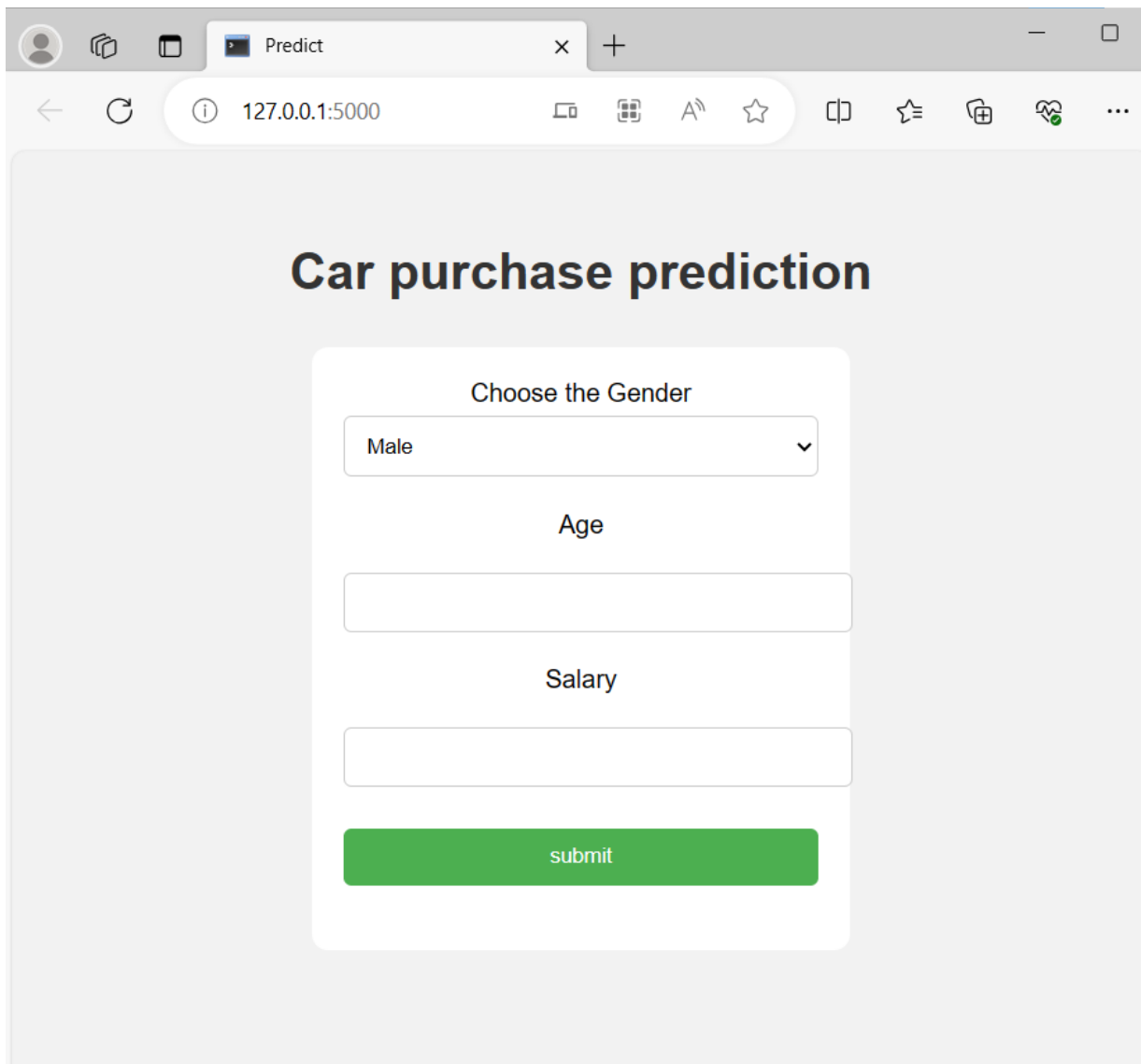


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
▼ TERMINAL
PS C:\Users\Sanmathi\Downloads\CarPurchasePrediction> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 218-447-671
```

When you run the app.py file and click on the server url in terminal, you will be redirected to home page.



**The index page will look like:**



The image shows a web browser window with a single tab titled 'Predict'. The address bar displays '127.0.0.1:5000'. The main content area has a light gray background and features a centered white form titled 'Car purchase prediction' in a large, bold, black font. The form contains three input fields: a dropdown menu for 'Choose the Gender' with 'Male' selected, a text input for 'Age', and another text input for 'Salary'. A green 'submit' button is positioned at the bottom of the form.

**Car purchase prediction**

Choose the Gender

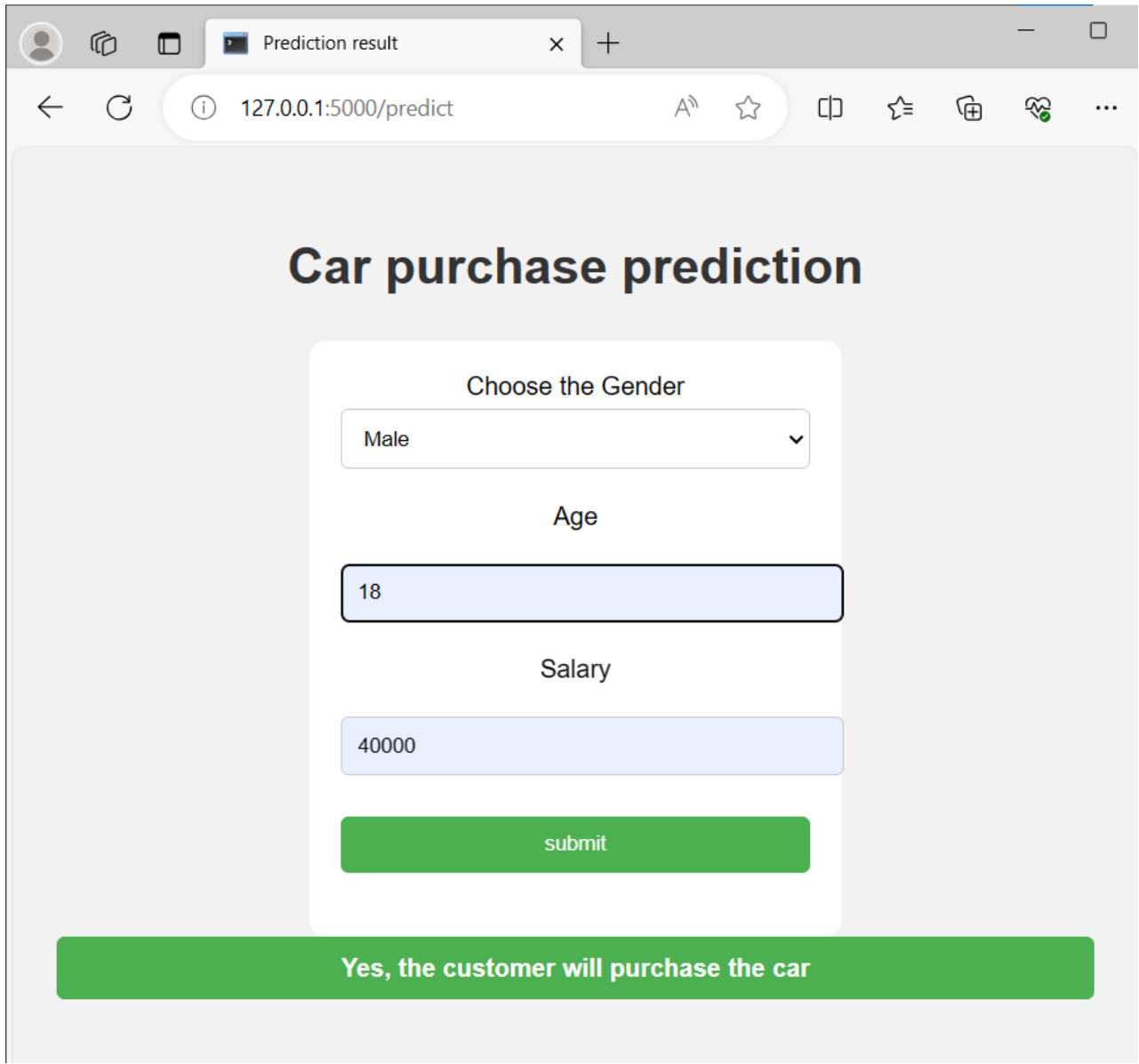
Male

Age

Salary

submit

If you click on submit button after entering the values you will be redirected to result.html page and it looks like:



The screenshot shows a web browser window with a single tab titled "Prediction result". The address bar displays "127.0.0.1:5000/predict". The page content features a large heading "Car purchase prediction" in bold black text. Below the heading is a white form box containing three input fields: a dropdown menu for "Choose the Gender" with "Male" selected, a text input for "Age" with "18" entered, and a text input for "Salary" with "40000" entered. A green "submit" button is positioned below these fields. At the bottom of the page, a wide green banner displays the prediction result: "Yes, the customer will purchase the car".

**Car purchase prediction**

Choose the Gender

Male

Age

18

Salary

40000

submit

**Yes, the customer will purchase the 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.