

# CAR PURCHASE PREDICTION USING ML

## PROJECT MANUAL

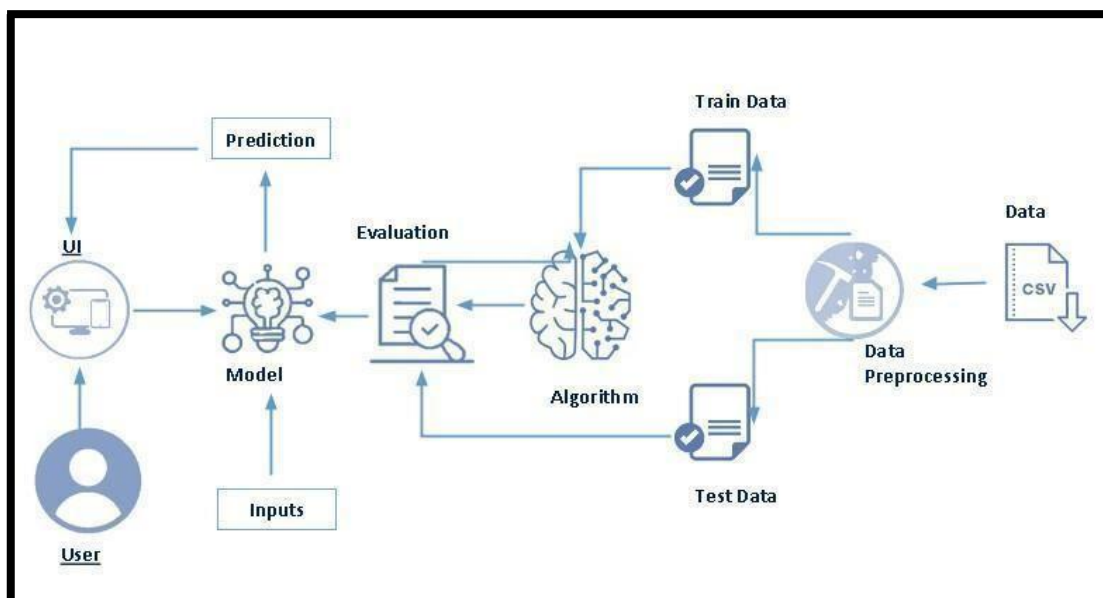
### Project Description:

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.

### Technical Architecture:



## Prerequisites:

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

- **Anaconda navigator and PyCharm:** ○ Refer the link below to download anaconda navigator  
○ Link: <https://youtu.be/1ra4zH2G4o0>
- **Python packages:**
  - Open anaconda prompt as administrator
  - 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**
  - Supervised learning: <https://www.javatpoint.com/supervised-machine-learning> ○  
Unsupervised learning: <https://www.javatpoint.com/unsupervised-machine-learning> ○  
SVM:  
<https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>
  - Decision Tree:  
<https://www.analyticsvidhya.com/blog/2022/03/decision-tree-machine-learning-using-python/>
  - Evaluation metrics:  
<https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-errormetrics/>
- **Flask Basics:** [https://www.youtube.com/watch?v=lj4I\\_CvBnt0](https://www.youtube.com/watch?v=lj4I_CvBnt0)

## 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:

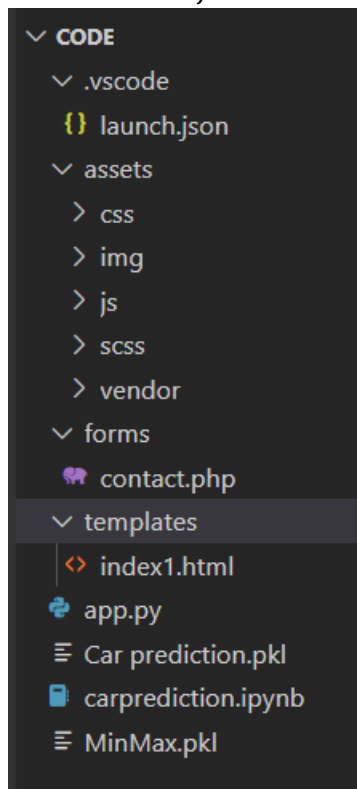
- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

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

- Data collection – Collect the dataset or create the dataset.
- Visualizing and analysing data – Univariate analysis, Bivariate analysis, Multivariate analysis and Descriptive analysis.
- Data pre-processing – Checking for null values, handling outliers, handling categorical data, and 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 and save the model.
- Application Building – Create an HTML file and 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 prediction.pkl is our saved model. Further we will use this model for flask integration.
- Templates folder contains html files and assets contains css,images,js,scss and vendor 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 behaviours, 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**

Here are some potential business requirements for Share price predictor.

**Data Collection and Integration:** Gather and integrate a comprehensive dataset that includes customer age and income information, ensuring data accuracy and consistency.

**Integration with Marketing Strategies:** The model should seamlessly integrate with the company's marketing strategies, enabling targeted campaigns towards potential car buyers.

**Scalability:** Design the model to handle a growing number of customer data points while maintaining prediction accuracy and response times.

**Resource Optimization:** Assist in optimizing marketing resources by directing efforts towards customers with a higher probability of purchasing a car, leading to reduced costs and improved ROI.

**Customization:** Consider the ability to customize the model for specific market segments or product lines, enabling fine-tuned predictions and targeted strategies.

## **Activity 3: Literature Survey**

A literature survey would involve researching and reviewing existing studies, articles, and other publications on the topic of project. The survey would aim to gather information on current systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous projects, and

any relevant data or findings that could inform the design and implementation of the current project.

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

The Anime Recommendation project can have both social and business impacts.

##### **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 analysing 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.

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

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.

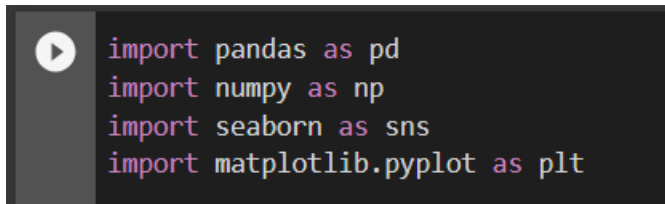
Link: <https://www.kaggle.com/code/vishesh1412/car-purchase-decision-eda-and-decision-tree/input>

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

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

### Activity 2: Importing the libraries.

Import the necessary libraries as shown in the image. Here we have used visualization style as five thirty-eight.

A screenshot of a code editor with a dark background. On the left, there is a play button icon. The code is written in a light-colored font and consists of four lines of import statements: 

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

### Activity 3: Read the Dataset

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

In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of 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

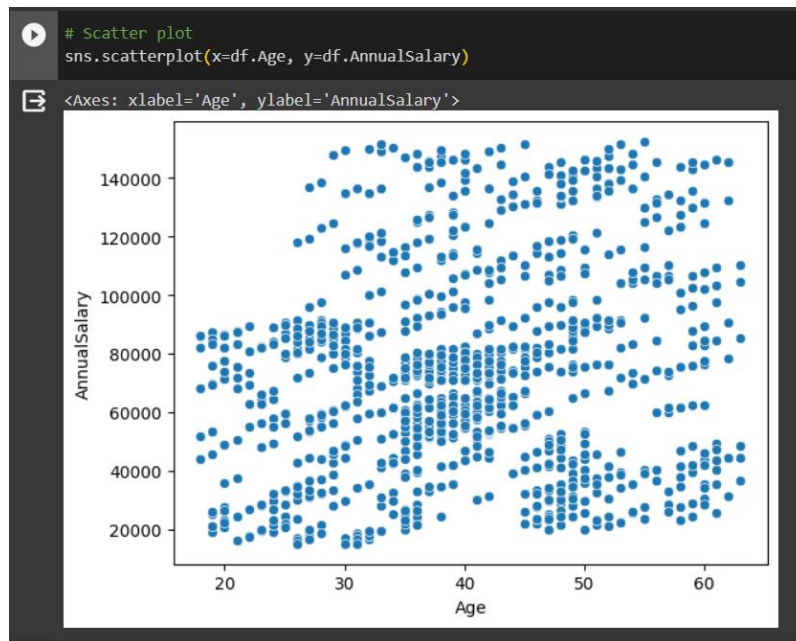
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Histplot and countplot.

- 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.



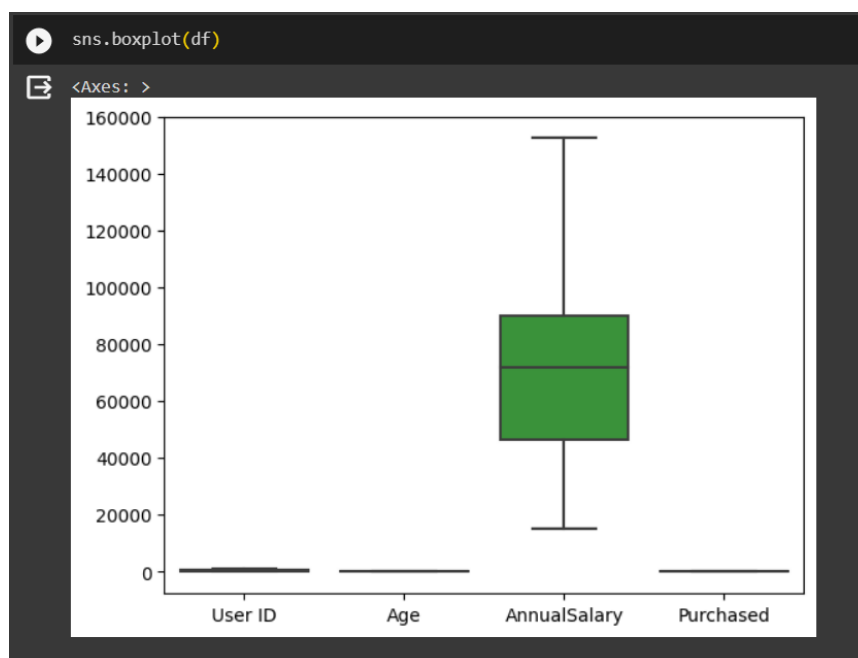
## Activity 5: Bivariate analysis

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



## Activity 6: Multivariate analysis

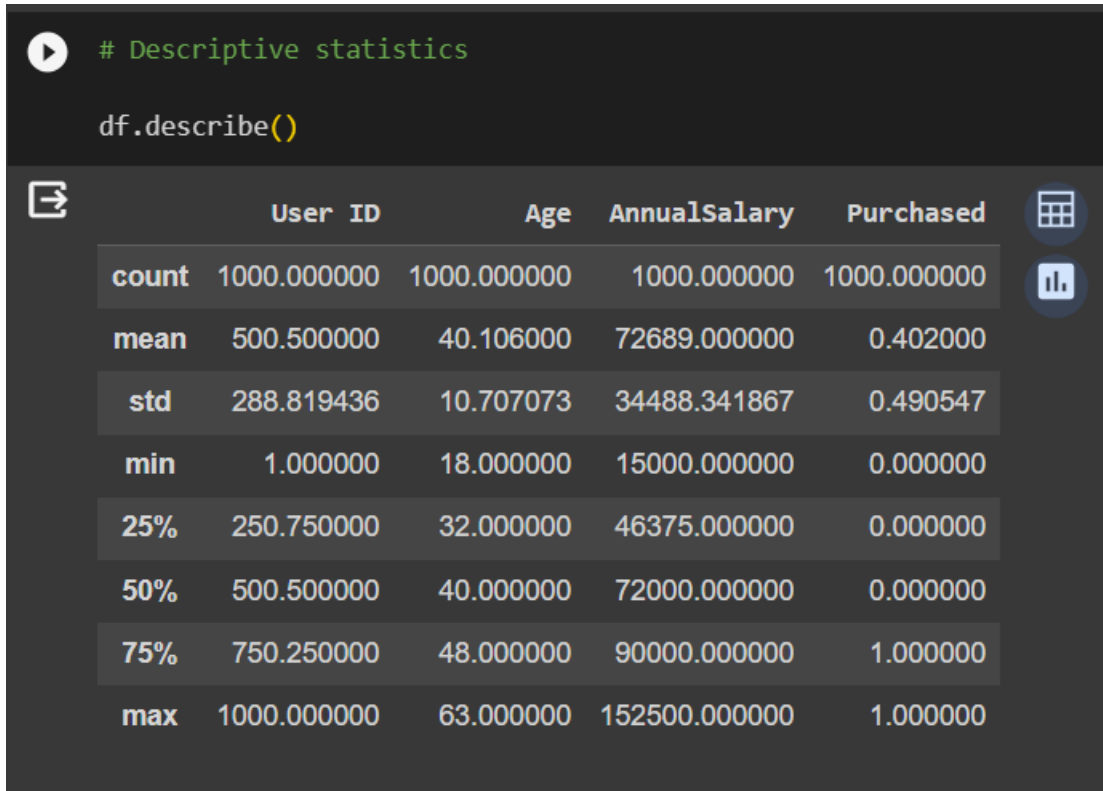
In simple words, multivariate analysis is to find the relation between multiple features. Here we have used boxplot from seaborn package.





## 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.



```
# Descriptive statistics
df.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

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

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.

```
df.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.

```
[6] # Checking for null values
df.isnull().any()
```

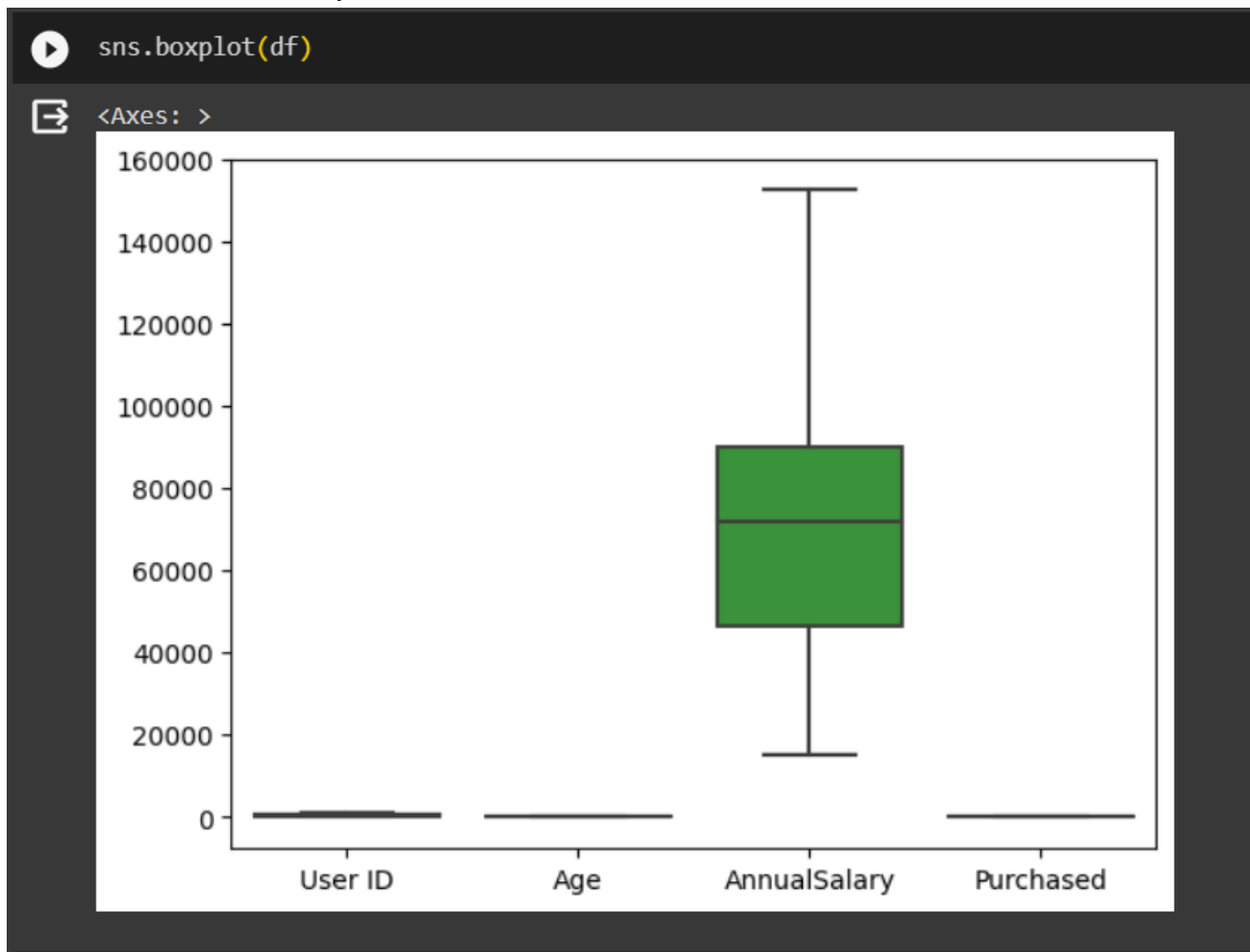
```
User ID      False
Gender       False
Age          False
AnnualSalary False
Purchased    False
dtype: bool
```

Let's look for any outliers in the dataset.

## 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.



## 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

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.

+

```
✓ [28] from sklearn.model_selection import train_test_split  
0s      x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.4,random_state=10)
```

## 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.

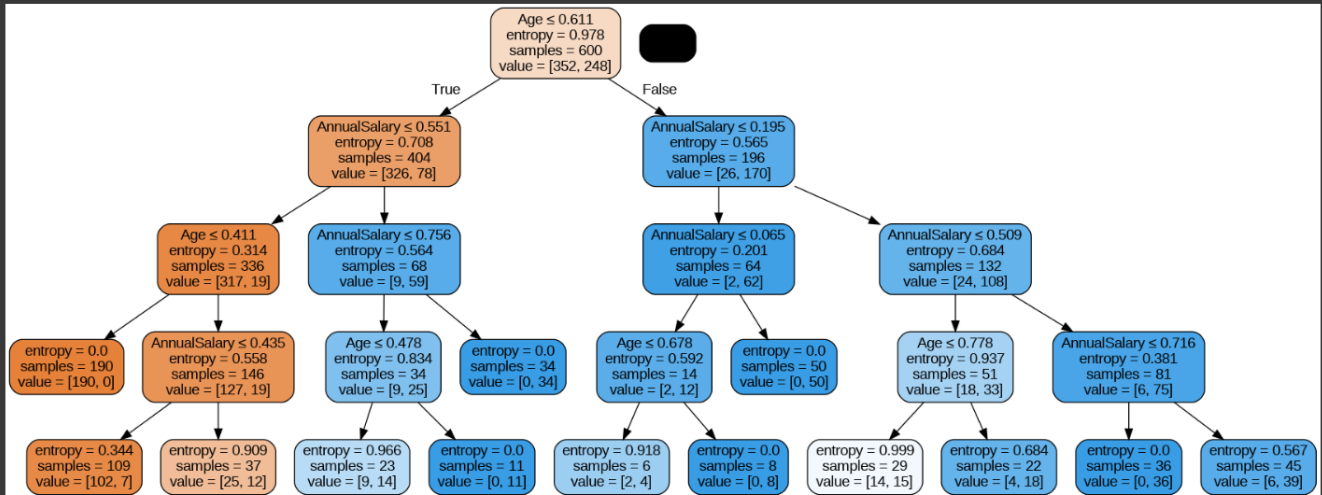
### Activity 1: Decision Tree Classifier

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 do

```
✓ [33] from sklearn.tree import DecisionTreeClassifier  
0s      model1 = DecisionTreeClassifier(max_depth=4,splitter='best',criterion='entropy')  
  
✓ 0s      model1.fit(X_train,Y_train)  
  
DecisionTreeClassifier  
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
[41] from six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import export_graphviz
```

```
[42] dot_data = StringIO()
export_graphviz(model1, out_file=dot_data, feature_names= X.columns,
                filled=True, rounded=True, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```



## Activity 2: Random Forest Classifier model

```
[44] from sklearn.ensemble import RandomForestClassifier
model2 = RandomForestClassifier(criterion='entropy')
```

```
[45] model2.fit(X_train, Y_train)
```

RandomForestClassifier  
RandomForestClassifier(criterion='entropy')

```
[46] r_y_predict = model2.predict(X_test)
r_y_predict_train = model2.predict(X_train)
```

```
[47] print('Testing Accuracy = ', accuracy_score(Y_test, r_y_predict))
print('Training Accuracy = ', accuracy_score(Y_train, r_y_predict_train))
```

Testing Accuracy = 0.8925  
Training Accuracy = 1.0

### Activity 3: Evaluating performance of the model

```
[50] # Use score method to get accuracy of model
score = model2.score(X_test, Y_test)
print("Accuracy using RFC: ",score)

# classification report
print(classification_report(Y_test,r_y_predict))
```

		precision	recall	f1-score	support
	0	0.95	0.87	0.91	246
	1	0.82	0.92	0.87	154
	accuracy			0.89	400
	macro avg	0.88	0.90	0.89	400
	weighted avg	0.90	0.89	0.89	400

```
[51] # Use confusion matrix
confusion_matrix(Y_test,r_y_predict)
```

```
array([[215,  31],
       [ 12, 142]])
```

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

```
[85] prediction = model2.predict((np.array([[0.384384,1.0,0.377778,0.036364]])))

if prediction[0] == 1:
    print("Purchased")
else:
    print("0")

0
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

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

```
import pickle

pickle.dump(DT_model,open('Car prediction.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 server side script

### Activity1: Building Html Pages:

For this project create HTML file namely

- index1.html save it in Templates folder.

```
<body>

<!-- ===== Header ===== -->
<header id="header">
  <div class="container">

    <h1><a href="index.html">CAR PURCHASE PREDICTION</a></h1>
    <!-- Uncomment below if you prefer to use an image logo -->
    <!-- <a href="index.html" class="mr-auto"></a> -->
    <h2>This website is designed to<span> predict the purchase</span> of a car by the customer.</h2>

    <nav id="navbar" class="navbar">
      <ul>
        <li><a class="nav-link active" href="#header">Home</a></li>
        <li><a class="nav-link" href="#about">About</a></li>
        <li><a class="nav-link" href="#pred">Prediction</a></li>
        <li><a class="nav-link" href="#team">The Team</a></li>
        <li><a class="nav-link" href="#contact">Contact</a></li>
      </ul>
      <i class="bi bi-list mobile-nav-toggle"></i>
    </nav><!-- .navbar -->

  </div>
</header><!-- End Header -->
```

```
127 <!-- ===== Prediction Section ===== -->
128 <section id="pred" class="pred">
129   <div class="container">
130
131     <div class="section-title">
132       <h2>Prediction</h2>
133       <p>Your Prediction</p>
134     </div>
135
136
137     <form action ="/login" method = "post">
138
139       <p>Age</p>
140       <p><input type ="text" name ="age"/></p>
141
142       <p>Annual Income</p>
143       <p><input type ="text" name ="annualincome"/></p>
144
145       <label for ="Gender">Gender</label>
146       <select name = "gender">
147         <option value =1>Male</option>
148         <option value =0>Female</option>
149
150       </select>
151
152       <p><input type ="submit" value ="submit"/></p>
153
154       <h2><b>{{output}}</b></h2>
155
```



## Activity 2: Build Python code:

Import the libraries

```
import numpy as np
from flask import Flask, request, render_template
import pickle
```

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 = Flask(__name__)

model = pickle.load(open('Car prediction.pkl','rb'))
```

Render HTML page:

```
@app.route('/')
def start():
    return render_template('index1.html')

@app.route('/login',methods =["POST","GET"])
def login():

    if request.method == "POST":
        age = request.form["age"]
        annual_income = request.form["annualincome"]
        gender = request.form["gender"]
```

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:

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.

```
t = np.array([[age,annual_income,gender]])

t_scaled= scale.fit_transform(t)
output =model.predict(t_scaled)
print(output)

if (output == 1):
    return render_template('index1.html', output="Purchasble")
else:
    return render_template('index1.html', output="Not purchasble")
    return render_template("index1.html")

return render_template("index1.html")
```

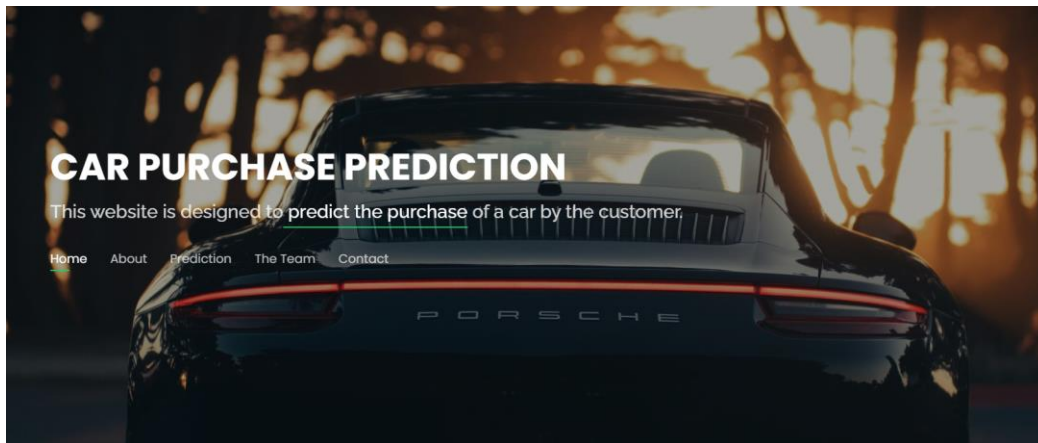
Main Function:

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

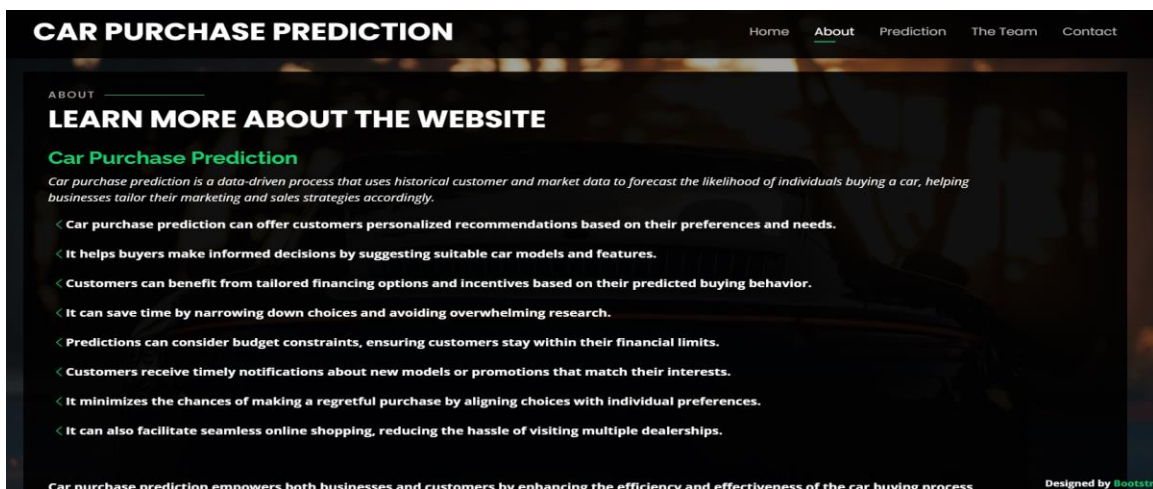
### Activity 3: Run the application

Open Visual studio code and Import all the project folders.

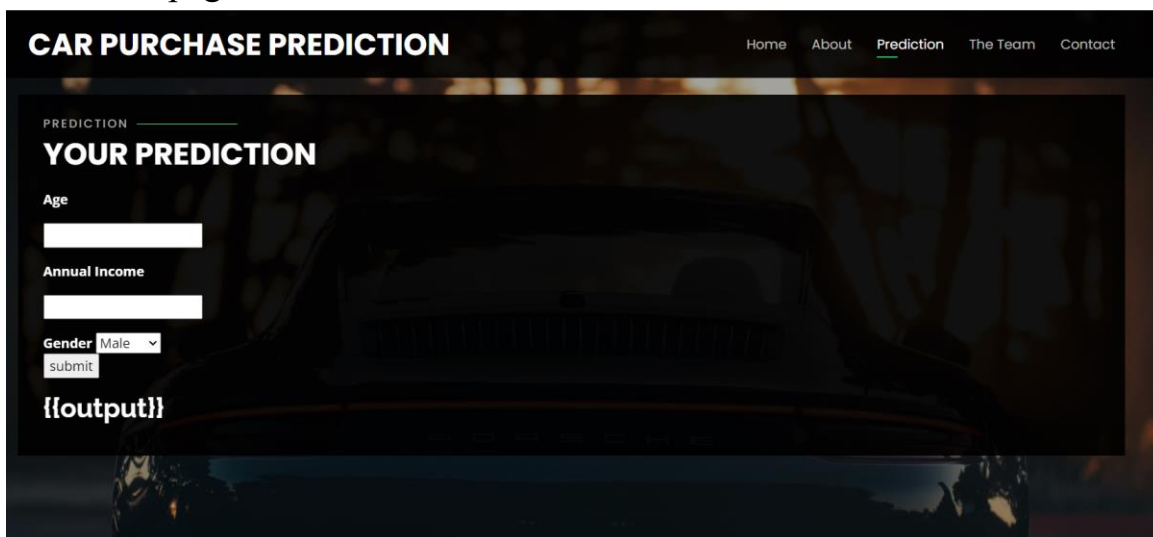
When you run the app.py file and click on the server url in terminal, you will be redirected to the home page. The home page will look like:



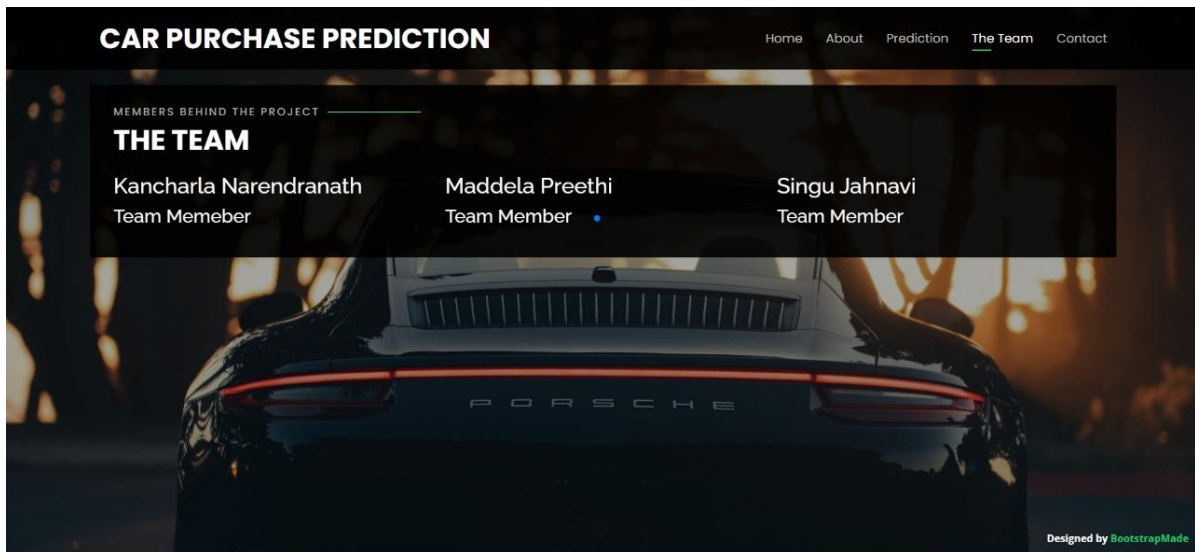
About:



Prediction page:



The team page:



## 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 behaviours. 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.