

Car purchase Prediction Using ML

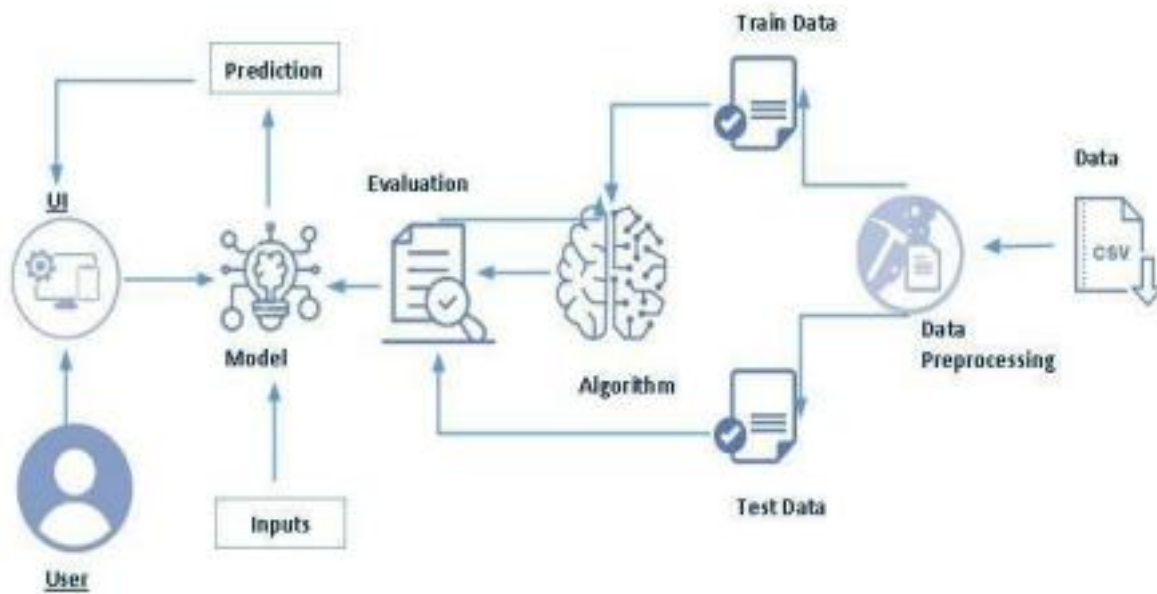
Project Description:

Created a novel machine learning (ML) system that uses convolutional neural networks (CNN) to forecast automobile purchases based on consumer information. benefited from the power of image-based data analysis to produce precise forecasts by leveraging characteristics like age, income, and past purchasing trends. By utilizing the visual cues found in images related to cars, this method achieved high prediction accuracy and added a distinctive element to the prediction process. The model provides rich visual insights to guide decision-making by estimating the likelihood that a potential buyer will make a purchase.

Made it possible for users to make simple predictions by seamlessly integrating the CNN model into an intuitive user interface. With the addition of image analysis, users can input images and pertinent demographic data to receive precise purchase likelihoods. Through the provision of insights for customized marketing strategies that make use of both textual and visual data, this project is revolutionizing the automotive industry. By enabling educated decisions and dealership targeting through a novel use of CNNs in auto purchase forecasts, it improves customer experiences.

By utilizing image-based data, the CNN model achieves a high accuracy rate through careful training and feature engineering, guaranteeing reliable predictions. Users input images and demographic data, which is seamlessly integrated into an intuitive interface, to obtain accurate purchase likelihoods with rich visual analysis. By enabling targeted customer engagement and resource optimization through the use of both textual and visual data insights, this innovation transforms marketing strategies and empowers automotive businesses to customize their approaches and increase overall efficiency.

Technical Architecture:



Pre requisites:

To complete this project, you must required following softwares, concepts and packages:

- **Anaconda Navigator:** You should have Anaconda Navigator installed to manage your Python environment and packages.
- **PyCharm (or any preferred Python IDE):** Choose a Python Integrated Development Environment (IDE) for coding and project development. This step is essential for creating, editing, and running your CNN-based car purchase prediction code.
- **Python packages:**
 - pip install numpy
 - pip install pandas
 - pip install scikit-learn
 - pip install matplotlib
 - pip install scipy
 - pip install pickle-mixin
 - pip install seaborn
 - pip install Flask

Prior Knowledge:

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

- **ML Concepts**
 - Supervised Learning: Understand the fundamentals of supervised machine learning, where models are trained on labeled data to make predictions.
 - Unsupervised Learning: Familiarize yourself with unsupervised machine learning, which deals with clustering and dimensionality reduction tasks without labeled data.
 - Convolutional Neural Networks (CNN): Gain a basic understanding of CNNs, as

- they are the primary focus of this project for image analysis and prediction.
-
- Decision Trees: While the primary focus is on CNNs, having a general understanding of decision trees is useful, as it's a common machine learning algorithm.
-
- Evaluation Metrics: Be familiar with common model evaluation metrics such as accuracy, precision, recall, and F1-score, as these will be used to assess the performance of your CNN model.
- **Flask Basics :**

Project Objectives:

By the end of this project you will:

- **Understand Fundamental Concepts of Convolutional Neural Networks (CNN):** You will gain a comprehensive understanding of how CNNs work, specifically for image-based data analysis and prediction, which is essential for car purchase prediction using visual cues.
- **Data Handling and Analysis:** Develop the skills to work with diverse data types, with a focus on image data, which will enable you to make informed decisions based on customer information and car images.
- **Data Preprocessing and Outlier Handling:** Acquire the knowledge and techniques required for data preprocessing and transformation, with a specific emphasis on image-related data. You will also learn how to handle outliers in image-based datasets.
- **Visualization Concepts for Image Data:** Gain insights into visualizing image data, which will enhance your ability to interpret and present results effectively to stakeholders.

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

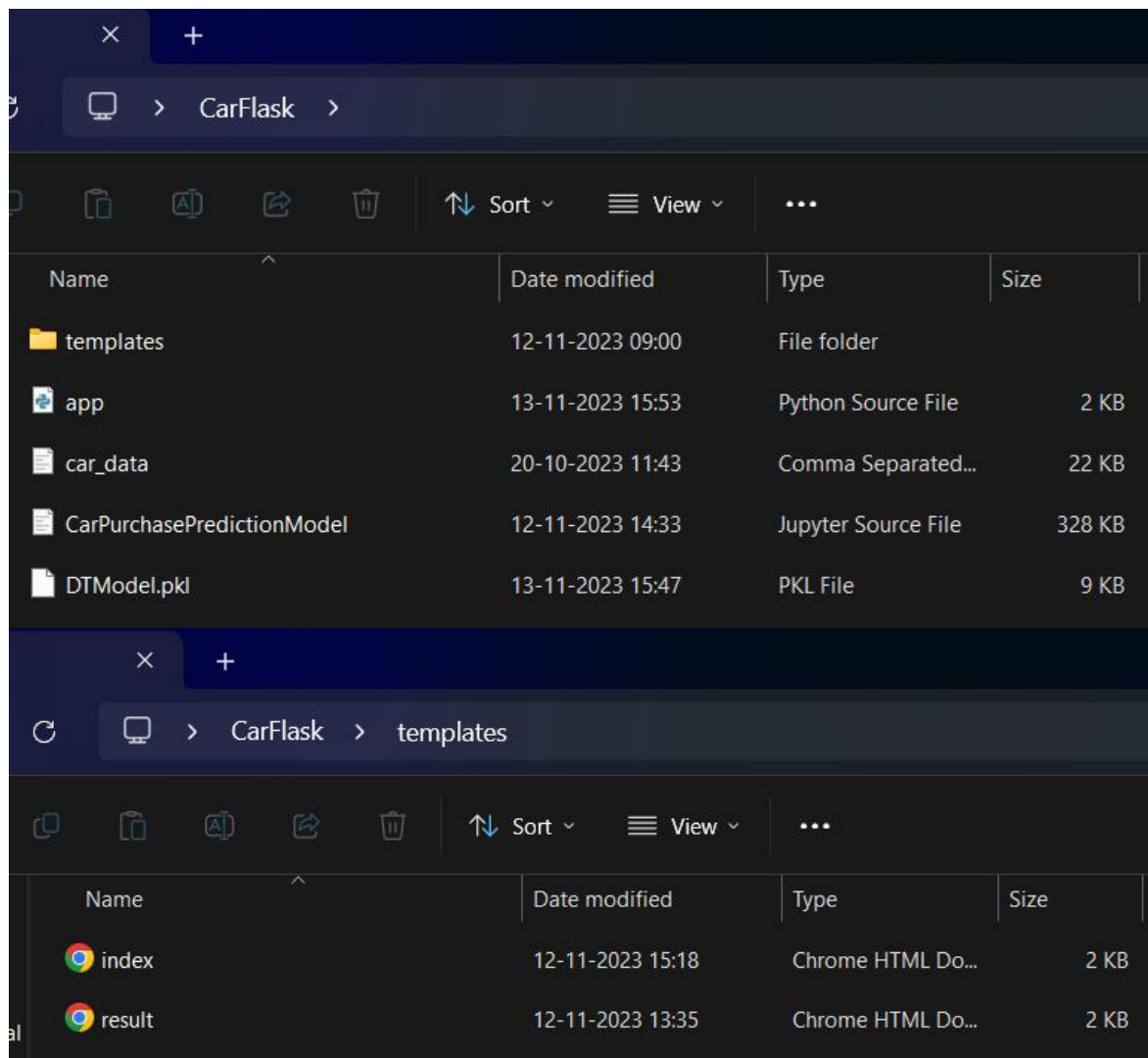
To accomplish this, we have to complete all the activities listed below,

- Data collection
 - Collect the dataset or create the dataset
- Visualizing and analyzing data
 - Univariate analysis
 - Bivariate analysis
 - Multivariate analysis
 - Descriptive analysis
- Data pre-processing
 - Checking for null values
 - Handling outlier
 - Handling categorical 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:

Created 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. 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 gender, 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

Here are some potential business requirements for Car Purchase 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 (Student Will Write)

The advent of online portals has enabled both buyers and sellers to access relevant information determining the market price of used cars. Machine learning algorithms, such as Lasso Regression, Multiple Regression, and Regression Trees, serve as examples. Our goal is to create a statistical model predicting the value of pre-owned vehicles using past customer data and various vehicle parameters. This project seeks to compare the predictive efficiency of different models to identify the most suitable one.

Numerous prior studies have delved into predicting used car prices. In Mauritius, Pudaruth employed methods like naive Bayes, k-nearest neighbors, multiple linear regression, and decision trees, but due to a limited number of observed cars, the predictive results were suboptimal. Pudaruth concluded that decision trees and naive Bayes are ineffective for continuous-valued variables.

Noor and Jan utilized multiple linear regression for price prediction, employing a variable selection method to identify influential variables and discarding the rest. Despite using only a few variables in the model, the linear regression achieved an outstanding 98 percent R-square.

Peerun et al. assessed the performance of neural networks in predicting used car prices, finding that support vector machine regression slightly outperformed neural networks and linear regression, especially on higher-priced cars. Various approaches, from multiple linear regression and k-nearest neighbor to naive Bayes, random forest, and decision tree, have been employed in the digital realm to accurately anticipate car prices.

In an attempt to facilitate user predictions, we developed a web application where users can assess the likelihood of a specific customer purchasing a car based on factors such as gender, age, and annual salary.

Activity 4: Social and Business Impact.

The Car Purchase Predictor 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 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

There are many popular open sources for collecting the data. Eg: 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.

```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC
from sklearn import preprocessing
import pickle
```

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.

Importing Dataset

```
In [ ]: df = pd.read_csv('car_data.csv')
df.head()
```

```
Out[4]:
```

	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

```
In [ ]: df.shape
```

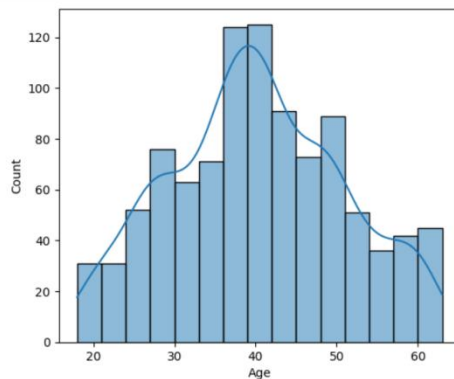
```
Out[5]: (1000, 5)
```

Activity 4: Univariate analysis

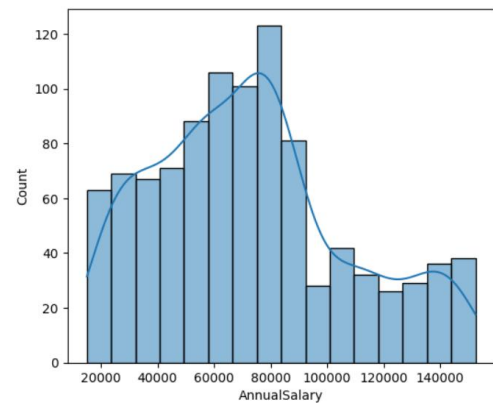
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as Histplot and countplot.

- 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
In [ ]: fig = plt.figure(figsize = (6, 5))
sns.histplot(data = df, x = 'Age', kde = True)
plt.show()
```



```
In [ ]: fig = plt.figure(figsize = (6, 5))
sns.histplot(data = df, x = 'AnnualSalary', kde = True)
plt.show()
```

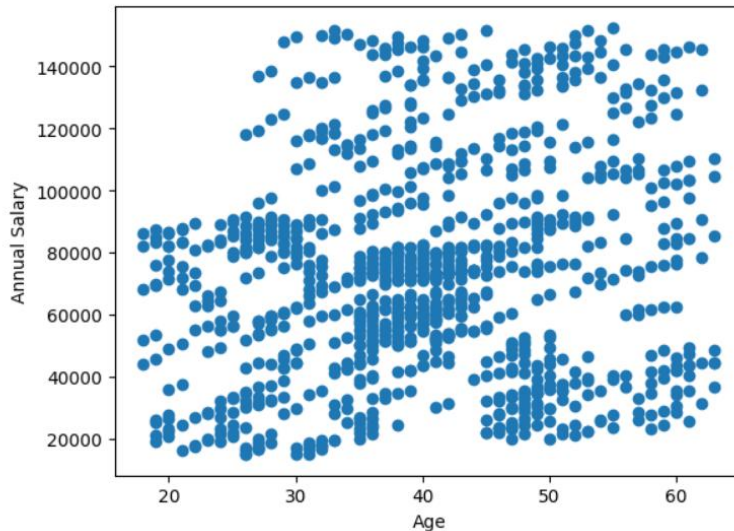


Activity 5: Bivariate analysis

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

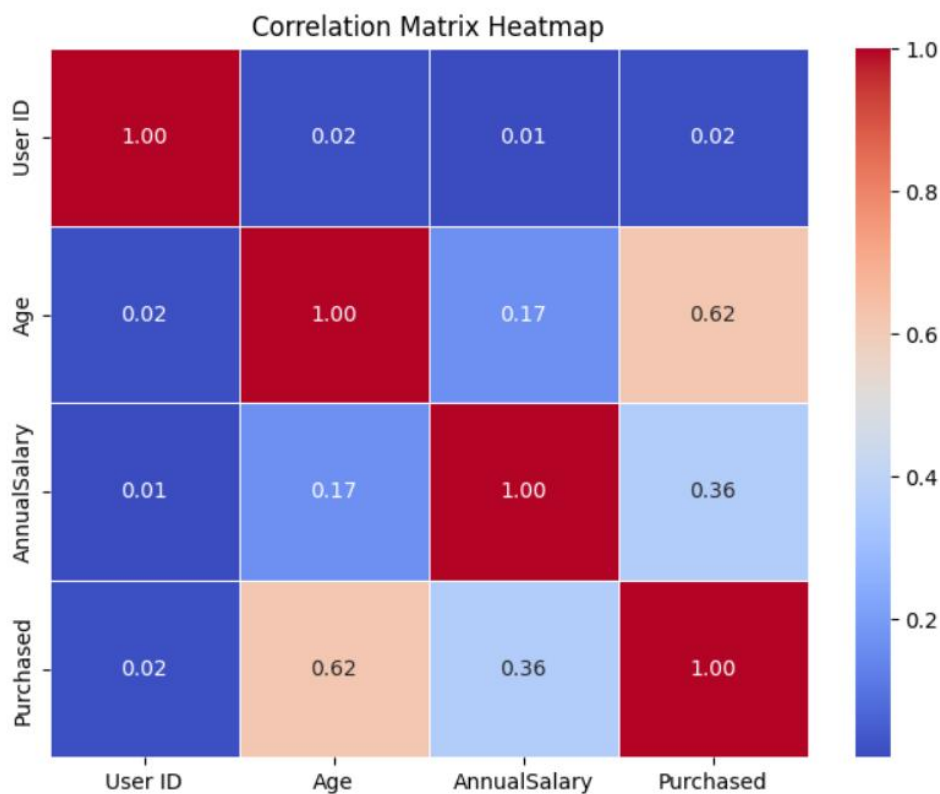
Bivariate Analysis

```
In [ ]: #age vs annualSalary
plt.scatter(data = df, x = 'Age', y = 'AnnualSalary')
plt.xlabel('Age')
plt.ylabel('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.



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 Analysis

```
In [ ]: df.describe()
```

```
Out[10]:
```

	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

- 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.

Handling Missing Values

```
In [ ]: df.isnull().sum()
```

```
Out[11]: User ID      0
        Gender      0
        Age         0
        AnnualSalary 0
        Purchased    0
        dtype: int64
```

Let's look for any outliers in the dataset

Activity 2: Handling outliers

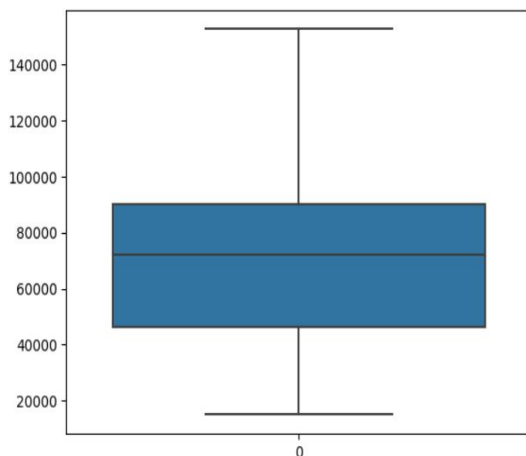
With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of all features with some mathematical formula.

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

Detecting Outliers

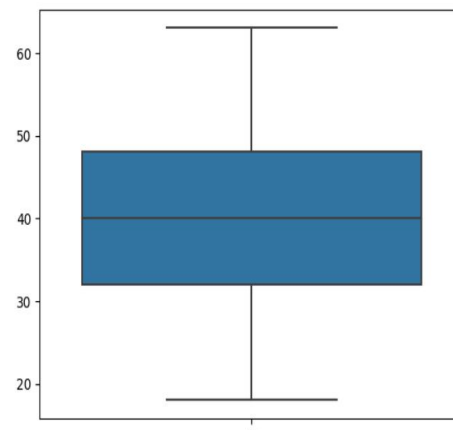
```
In [ ]: sns.boxplot(data = df['AnnualSalary'])
```

```
Out[12]: <Axes: >
```



```
In [ ]: sns.boxplot(data = df['Age'])
```

```
Out[13]: <Axes: >
```



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.

Splitting in Train and Test Data

```
In [ ]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
print('Train Data Size: ', X_train.shape)
print('Test Data Size: ', X_test.shape)
```

Train Data Size: (800, 4)

Test Data Size: (200, 4)

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 done.

```
#importing libraries
from sklearn.tree import DecisionTreeClassifier
import sklearn.tree as tree
from sklearn.model_selection import cross_val_score

#using GridSearchCV to find out the best parameters
from sklearn.model_selection import GridSearchCV
dt_classifier = DecisionTreeClassifier()

param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

grid_search = GridSearchCV(dt_classifier, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

print("Best Hyperparameters:", grid_search.best_params_)

best_dt_classifier = grid_search.best_estimator_
accuracy = best_dt_classifier.score(X_test, y_test)
print(f"Test Accuracy with Best Hyperparameters: {accuracy:.2f}")

Best Hyperparameters: {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
Test Accuracy with Best Hyperparameters: 0.91
```

```
[ ] #building the model

model1 = DecisionTreeClassifier(criterion = 'entropy', max_depth = None, max_features = 'log2', min_samples_leaf = 1, min_samples_split = 10, splitter = 'best')
model1
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_features='log2',
min_samples_split=10)
```

```
[ ] # Perform 5-fold cross-validation
cross_val_scores = cross_val_score(model1, X, y, cv=10)

# Print the cross-validation scores
print("Cross-validation scores:", cross_val_scores)

# Calculate and print the average accuracy across all folds
print("Average accuracy: {:.2f}".format(cross_val_scores.mean()))

Cross-validation scores: [0.87 0.85 0.89 0.87 0.9 0.91 0.86 0.92 0.85 0.87]
Average accuracy: 0.88
```

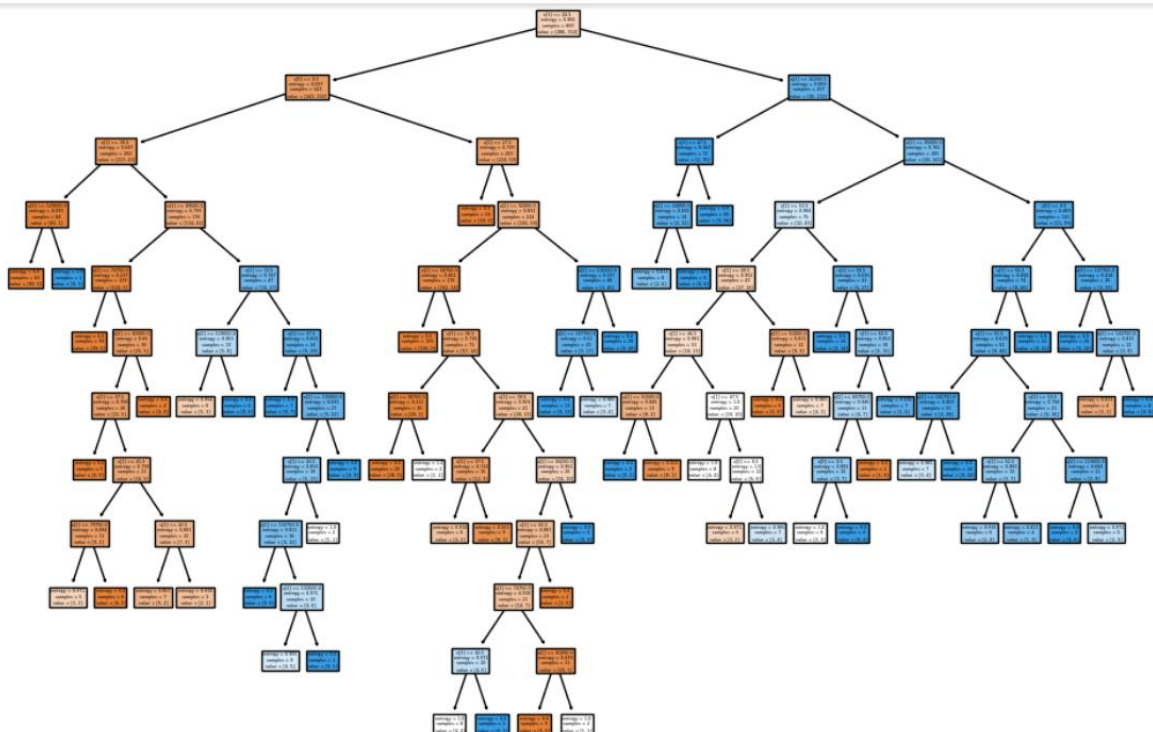
▶ #fitting the model

```
model1.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_features='log2',
min_samples_split=10)
```

```
[ ] #predicting for both train and test sets
model1pred_train = model1.predict(X_train)
model1pred_test = model1.predict(X_test)
```

```
[ ] #visualizing the decision tree
fig, ax = plt.subplots(figsize=(12, 8))
tree.plot_tree(model1, filled=True, ax=ax)
mplcursors.cursor(hover=True)
mplcursors.cursor(ax, multiple=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. For evaluating the model, confusion matrix and classification report is done.

SVC Classifier

```
In [ ]: #importing libraries
        from sklearn.svm import SVC
```

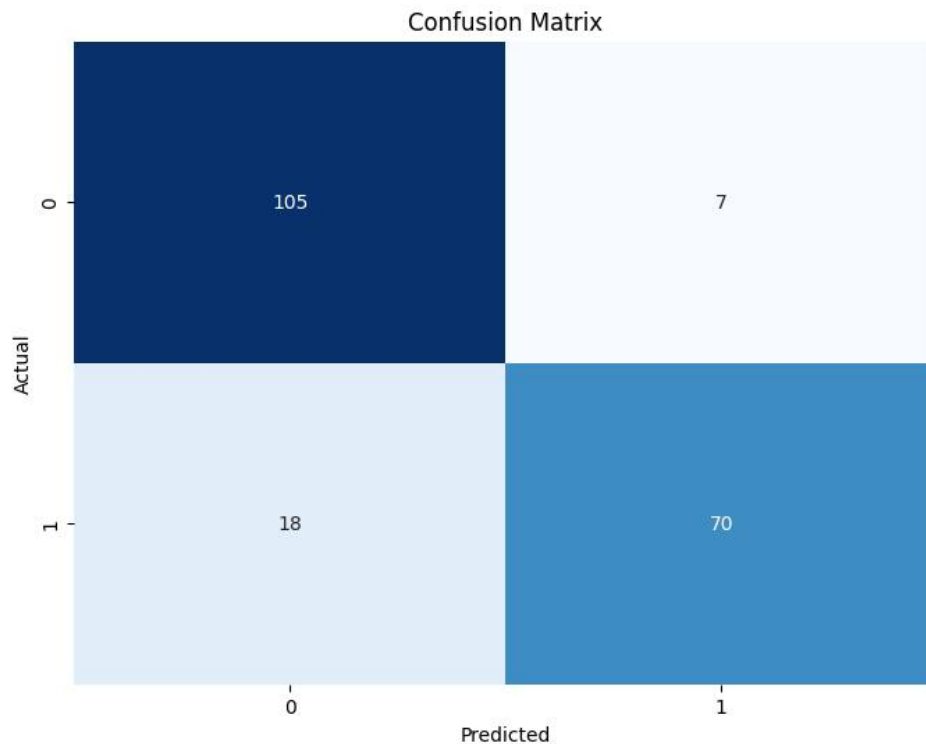
```
In [ ]: #building the model

        model2 = SVC(kernel = 'linear', random_state = 42)
        model2.fit(X_train, y_train)
```

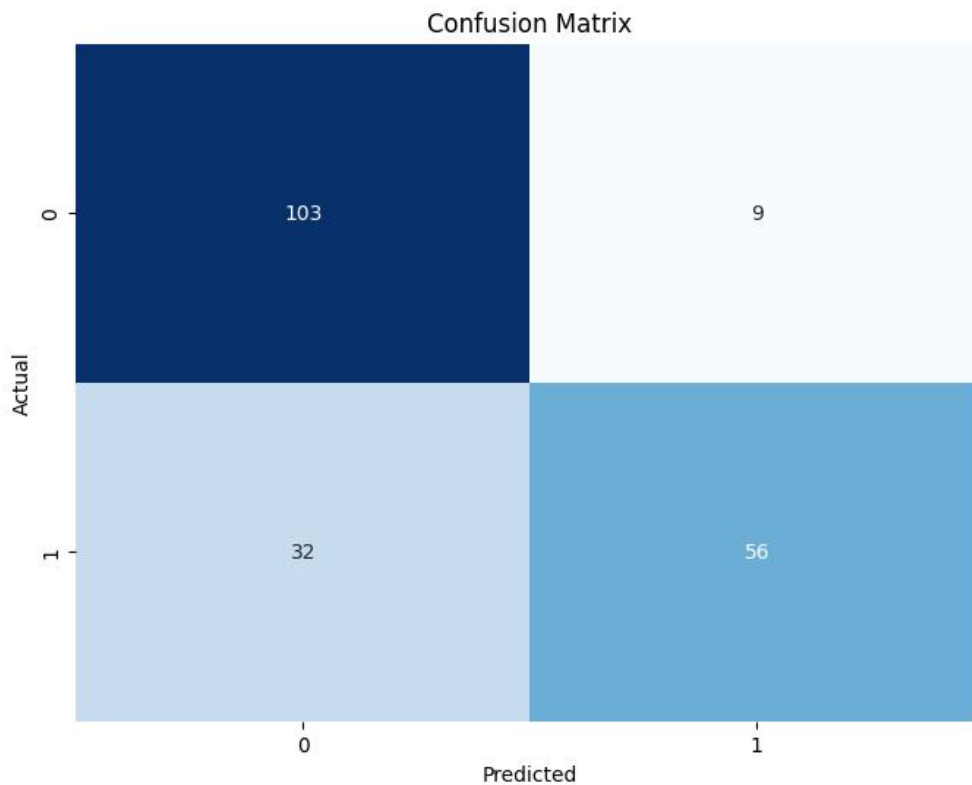
```
Out[27]: SVC(kernel='linear', random_state=42)
```

Activity 3: Evaluating performance of the model

DT Classifier:



SVC Classifier:



Activity 4: Saving The Model

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

Saving the Model

```
In [ ]: #libraries
import pickle
```

```
In [3]: #dumping model
pickle.dump(model1, open('/content/DTModel', '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

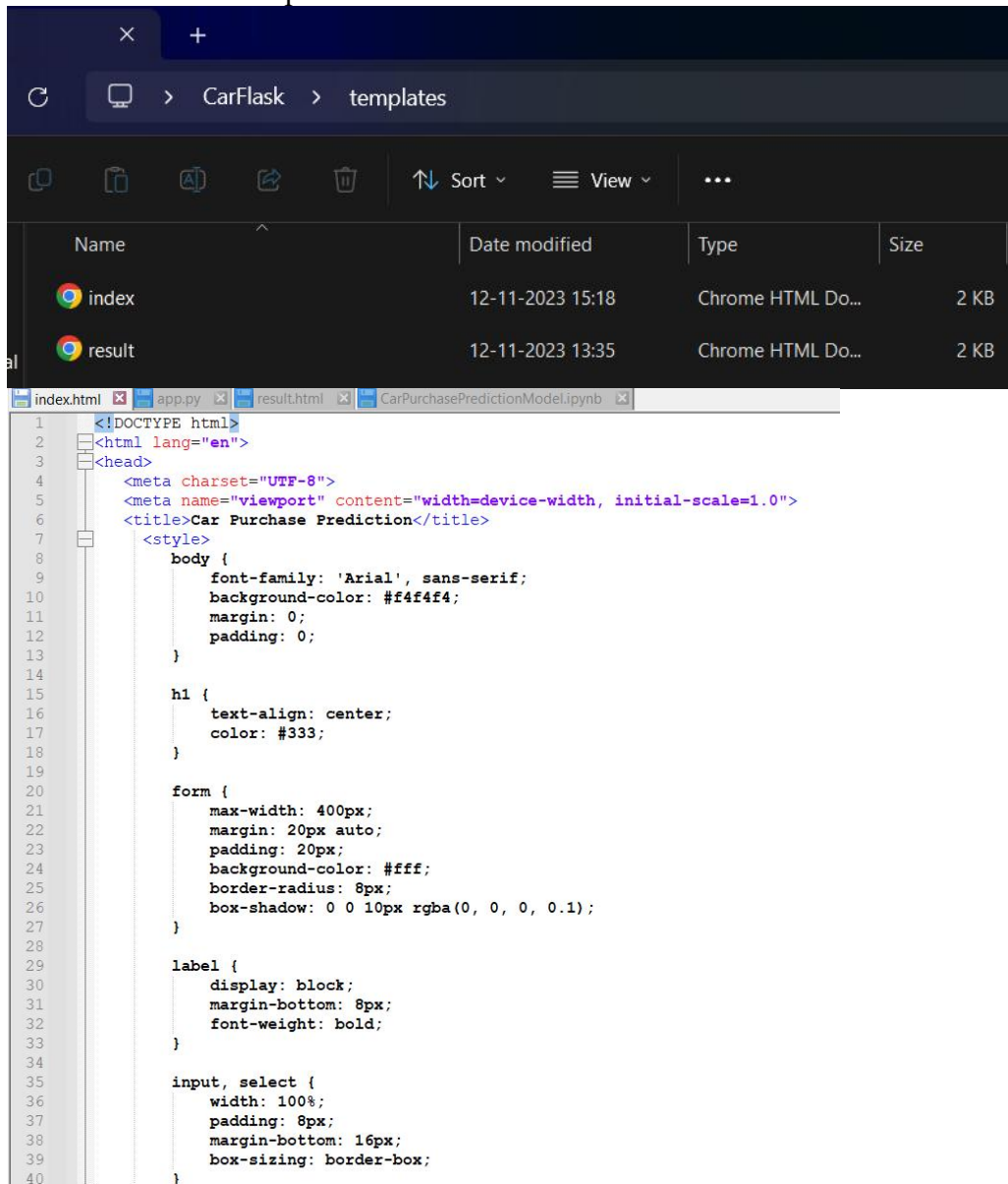
- Building HTML Pages
- Building serverside script

Activity1: Building Html Pages:

For this project we will create two HTML files, namely:

- index.html
- result.html

and save them in Templates folder.




```

41
42     input[type="submit"] {
43         background-color: #4caf50;
44         color: white;
45         cursor: pointer;
46     }
47
48     input[type="submit"]:hover {
49         background-color: #45a049;
50     }
51 </style>
52 </head>
53 <body>
54     <h1>Car Purchase Prediction</h1>
55     <form action="/predict" method="post">
56         Age: <input type="text" name="age" required><br>
57         Gender: <select name="gender" required>
58             <option value="Male">Male</option>
59             <option value="Female">Female</option>
60         </select><br>
61         Salary: <input type="text" name="salary" required><br>
62         <input type="submit" value="Predict">
63     </form>
64
65 </body>
66 </html>

```

index.html x app.py x result.html x CarPurchasePredictionModel.ipynb x

```

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>Prediction Result</title>
7     <style>
8         body {
9             font-family: 'Arial', sans-serif;
10            background-color: #f4f4f4;
11            margin: 0;
12            padding: 0;
13        }
14
15        h1 {
16            text-align: center;
17            color: #333;
18        }
19
20        p {
21            max-width: 400px;
22            margin: 20px auto;
23            padding: 20px;
24            background-color: #fff;
25            border-radius: 8px;
26            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
27        }
28
29        a {
30            display: block;
31            text-align: center;
32            margin-top: 20px;
33            text-decoration: none;
34            color: #4caf50;
35        }
36
37        a:hover {
38            color: #45a049;
39        }
40    </style>
41 </head>

```

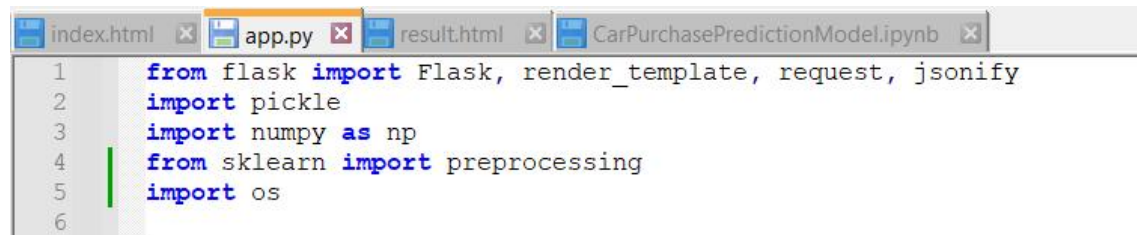
```

42     <body>
43         <h1>Prediction Result</h1>
44         <p>{{ result }}</p>
45         <a href="/">Go back</a>
46     </body>
47 </html>
48

```

Activity 2: Build Python code:

Import the libraries

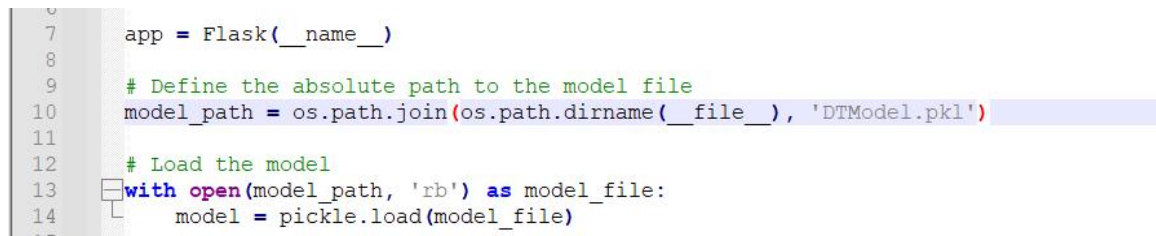


```

1  from flask import Flask, render_template, request, jsonify
2  import pickle
3  import numpy as np
4  from sklearn import preprocessing
5  import os
6

```

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.

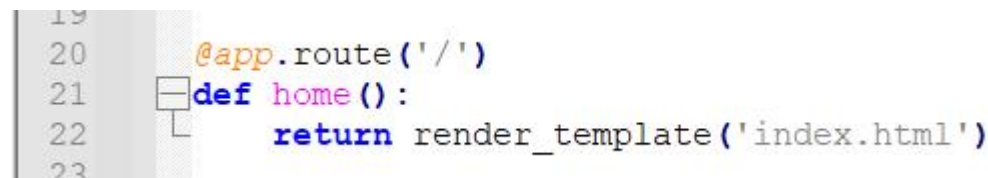


```

7  app = Flask(__name__)
8
9  # Define the absolute path to the model file
10 model_path = os.path.join(os.path.dirname(__file__), 'DTModel.pkl')
11
12 # Load the model
13 with open(model_path, 'rb') as model_file:
14     model = pickle.load(model_file)
15

```

Render HTML page:



```

19
20 @app.route('/')
21 def home():
22     return render_template('index.html')
23

```

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:

```

24 @app.route('/predict', methods=['POST'])
25 def predict():
26     # Get user input from the form
27     age = float(request.form['age'])
28     gender = request.form['gender']
29     salary = float(request.form['salary'])
30
31     # Encode gender using the LabelEncoder
32     gender_encoded = le_Sex.transform([gender])[0]
33
34     # Create a NumPy array with the input values
35     input_data = np.array([[age, gender_encoded, salary]])
36
37     # Make the prediction
38     prediction = model.predict(input_data)
39
40     # Display the prediction
41     result = "Will Buy a Car" if prediction[0] > 0.5 else "Will Not Buy a Car"
42

```

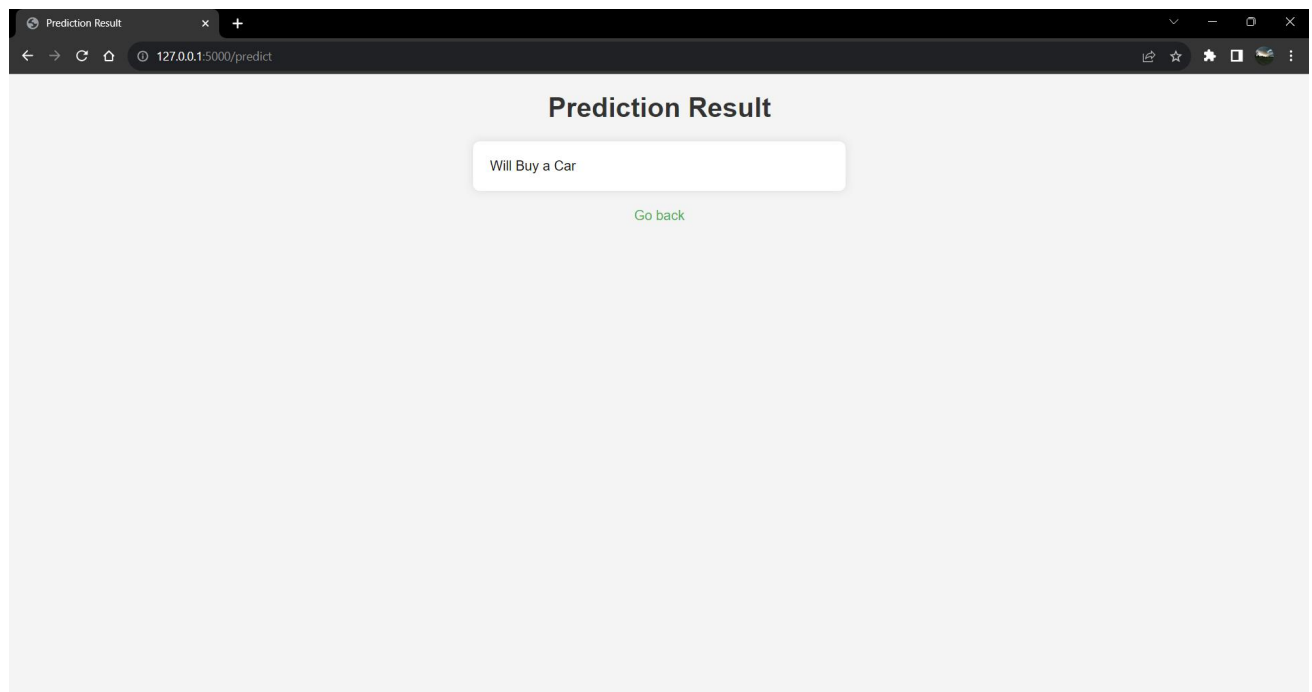
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text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

Activity 3: Run the application

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

The screenshot shows a web browser window titled "Car Purchase Prediction". The address bar displays "127.0.0.1:5000". The main content area features a form with the title "Car Purchase Prediction". The form contains three input fields: "Age:" (a text input), "Gender:" (a dropdown menu with "Male" selected), and "Salary:" (a text input). Below these fields is a green button labeled "Predict".

If you enter the values and click predict, it looks like:



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.