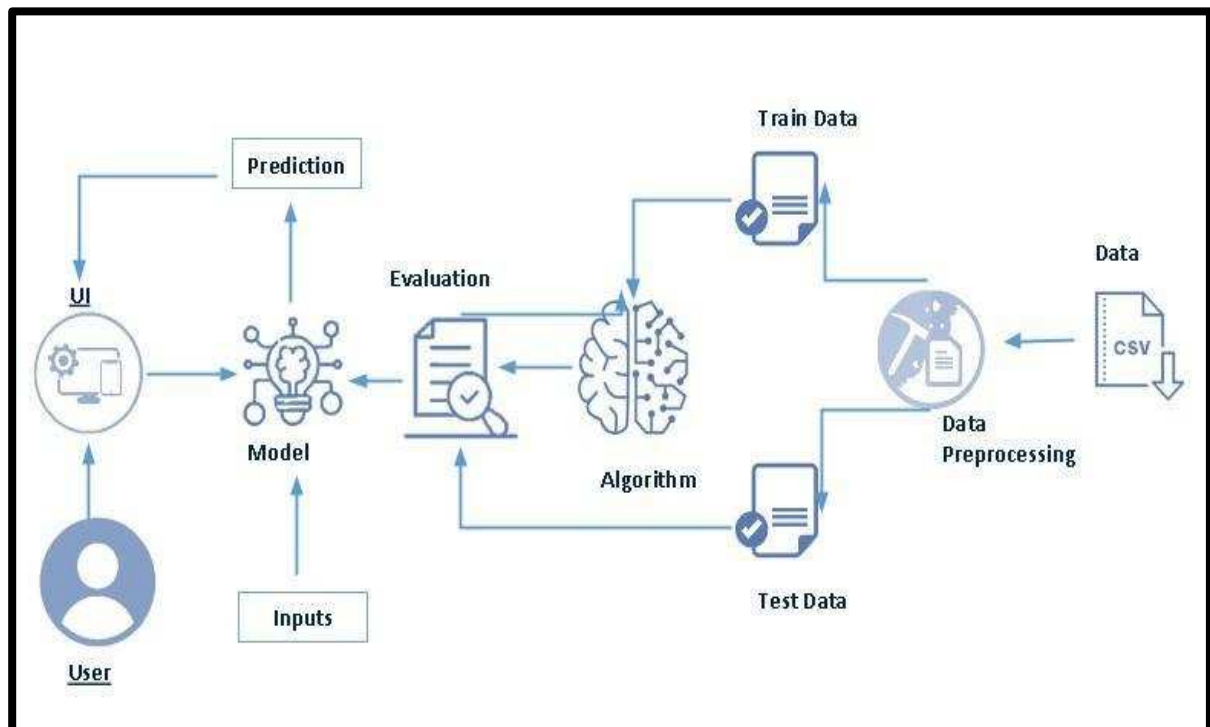


Wholesale Customer Segmentation Analysis Using ML

Project Description:

- This project aims to analyse the spending behaviour of wholesale customers and identify opportunities for growth. The data set consists of annual spending (in monetary units) on various product categories, including fresh, milk, grocery, frozen, detergents and paper, and delicatessen products. Additionally, the data includes information on the channel (hotel/restaurant/cafe or retail) and region (Lisbon, Oporto, or other) of the customer.
- By identifying customer segments with distinct spending behaviours, the project aims to provide insights on how wholesale businesses can tailor their marketing strategies and product offerings to better serve each customer segment.
- The outcomes of this project can have several applications. Business can utilize the predictive model to identify customers with distinct spending behaviours.

Technical Architecture:



Pre requisites:

To complete this project, you must required 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>
 -
 - KNN: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
 - Logistic Regression: <https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-logistic-regression/>
 - Evaluation metrics: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/>
- **Flask Basics** : https://www.youtube.com/watch?v=Ij4I_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:

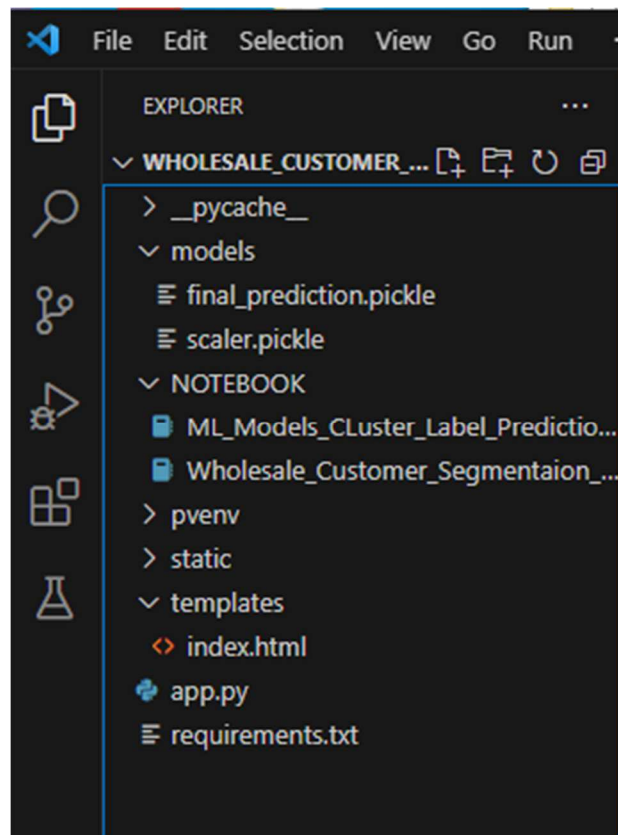
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

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:

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. Further we will use this model for flask integration.

Milestone 1 : Define Problem/Problem Understanding

Activity 1: Specify the business problem

- The Wholesale Customer Segmentation project aims to analyze the spending behaviour of wholesale customers and identify opportunities for growth. The data set consists of annual spending (in monetary units) on various product categories, including fresh, milk, grocery, frozen, detergents and paper, and delicatessen products. Additionally, the data includes information on the channel (hotel/restaurant/café or retail) and region (Lisbon, Oporto, or other) of the customer.
- Using unsupervised machine learning techniques, specifically clustering algorithms, the project seeks to group customers with similar spending patterns together. By identifying customer segments with distinct spending behaviours, the project aims to provide insights on how wholesale businesses can tailor their marketing strategies and product offerings to

better serve each customer segment. The project also aims to identify opportunities for growth, such as which products or product categories are underrepresented among customers, and which segments may be receptive to new product offerings.

- Overall, the Wholesale Customer Segmentation project seeks to provide valuable insights for wholesale businesses on how to optimize their operations and increase customer satisfaction and retention.

Activity 2: Business requirements

Here are some potential business requirements for Wholesale Customer Segmentation.

Accurate forecasting: The predictor must be able to accurately forecast the spending behaviour of wholesale customers.

User-friendly interface: The predictor must have a user-friendly interface that is easy to navigate and understand. The interface should present the results of the predictor in a clear and concise manner to provide valuable insights for wholesale businesses on how to optimize their operations and increase customer satisfaction and retention.

Activity 3: Literature Survey

Wholesale customer segmentation is the process of dividing wholesale customers into groups based on their shared characteristics, such as spending habits, location, or industry. This can be a valuable tool for wholesale businesses to better understand their customers and tailor their marketing and sales strategies accordingly.

There is a growing body of literature on wholesale customer segmentation. A 2019 study by the Aberdeen Group found that businesses that use customer segmentation are more likely to achieve their revenue and profit goals than those that do not. The study also found that businesses that use customer segmentation are better able to:

Target their marketing campaigns more effectively
Develop products and services that meet the needs of their customers
Increase customer satisfaction and retention

There are a number of different ways to segment wholesale customers. Some common methods include:

- **Geographic segmentation:** This involves dividing customers into groups based on their location. This can be a useful way to target customers with local marketing campaigns or to tailor product offerings to meet the needs of customers in different regions.

- **Demographic segmentation:** This involves dividing customers into groups based on their age, gender, income, or other demographic characteristics. This can be a useful way to target customers with specific products or services.
- **Behavioral segmentation:** This involves dividing customers into groups based on their buying habits, such as the products they purchase, the frequency of their purchases, or the amount they spend. This can be a useful way to identify customers who are most likely to respond to a particular marketing campaign or to develop new products or services that meet the needs of these customers.

Wholesale customer segmentation can be a valuable tool for businesses of all sizes. By understanding their customers and their needs, businesses can better tailor their marketing and sales strategies to achieve their goals.

Activity 4: Social or Business Impact.

The social and business impact of the Wholesale Customer Segmentation project are as follows:

- **Increased customer satisfaction and retention:** By understanding the spending behavior of their customers, wholesale businesses can tailor their marketing strategies and product offerings to better meet the needs of each customer segment. This can lead to increased customer satisfaction and retention, as customers are more likely to do business with companies that understand their needs and preferences.
- **Improved operational efficiency:** By identifying opportunities for growth, such as which products or product categories are underrepresented among customers, and which segments may be receptive to new product offerings, wholesale businesses can improve their operational efficiency. This can be done by streamlining their supply chain, optimizing their inventory management, and allocating resources more effectively.
- **Increased profitability:** By improving customer satisfaction and retention, and by improving operational efficiency, wholesale businesses can increase their profitability. This can be done by generating more revenue from existing customers, by acquiring new customers, and by reducing costs.

Milestone 2: Data Collection

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 The Wholesale Customer Segmentation data. This data is downloaded from kaggle.com. Please refer the link given below to download the dataset.

Link: <https://www.kaggle.com/datasets/binovi/wholesale-customers-data-set>

Milestone 3: Visualizing and analysing the data

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 1: Importing the libraries

Import the necessary libraries as shown in the image.

Activity 2: 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 the libraries
import numpy as np
import pandas as pd
from numpy import math
import seaborn as sns
from datetime import datetime

import warnings
from pylab import rcParams
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
warnings.filterwarnings('ignore')

# Load The Dataset
df = pd.read_csv("/content/drive/MyDrive/DATA/Wholesale customers data.csv")
df.head()

```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

Activity 3: Univariate analysis

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

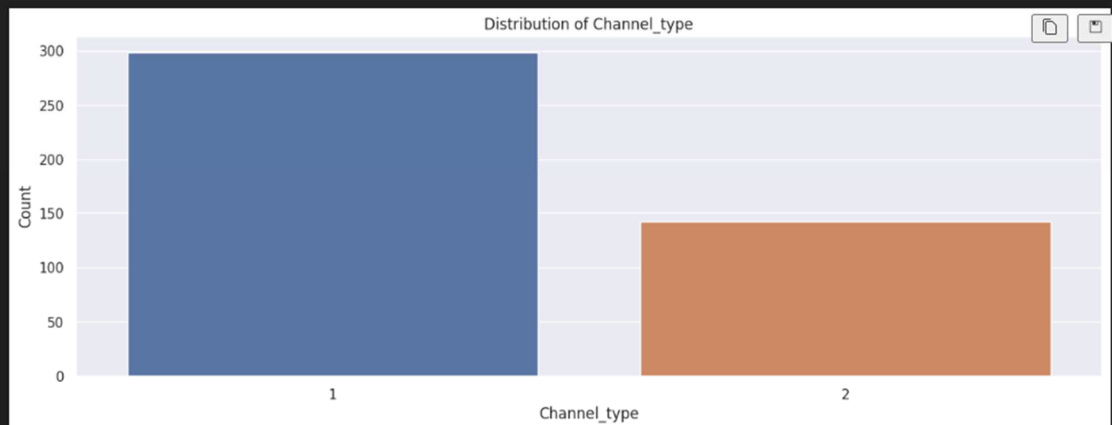

```
# Channel wise values count
Channel_df=df['Channel'].value_counts().reset_index()
Channel_df.rename(columns={'index': 'Channel_type'}, inplace=True)
Channel_df.rename(columns={'Channel': 'Count'}, inplace=True)
Channel_df.head()
```

	Channel_type	Count
0	1	298
1	2	142

```
plt.figure(figsize=(15,5))
plt.title('Distribution of Channel_type')
sns.barplot(x='Channel_type',y='Count',data=Channel_df)
```

Python

<Axes: title={'center': 'Distribution of Channel_type'}, xlabel='Channel_type', ylabel='Count'>



Activity 4: Bivariate analysis

To find the relation between two features we use bivariate analysis.

Activity 5: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features.

Activity 6: Descriptive analysis

```
df.describe()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

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.

Milestone 4: 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.
- 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.

Let's look for any outliers in the dataset

```
[ ] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype  
---  -
 0   gender              1000 non-null   object  
 1   race/ethnicity       1000 non-null   object  
 2   parental level of education 1000 non-null   object  
 3   lunch               1000 non-null   object  
 4   test preparation course 1000 non-null   object  
 5   math score          1000 non-null   int64   
 6   reading score       1000 non-null   int64   
 7   writing score        1000 non-null   int64   
dtypes: int64(3), object(5)
memory usage: 62.6+ KB

Checking whether the null values present or not

[ ] df.isnull().sum()

gender              0
race/ethnicity      0
parental level of education 0
lunch               0
test preparation course 0
math score          0
reading score       0
writing score       0
dtype: int64

[ ] df.isnull().sum().sum()

0
```

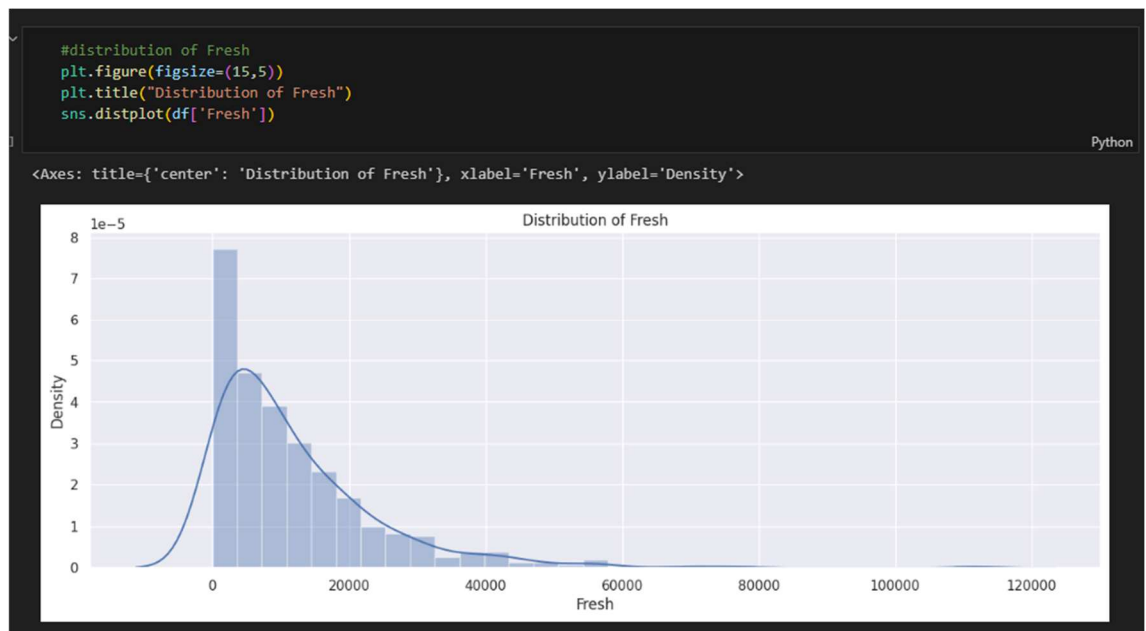
Activity 2: Handling outliers

With the help distribution, 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 Distribution of feature has outliers or Not.

```
df.Fresh.describe([.75,.90,.95,.99])

count      440.000000
mean       12000.297727
std        12647.328865
min         3.000000
50%        8504.000000
75%        16933.750000
90%        27090.500000
95%        36818.500000
99%        56082.610000
max        112151.000000
Name: Fresh, dtype: float64
```



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.

```
Train ,Test and Cross-Validation Dataset Construction

# split the data into test and train by maintaining same distribution of output variable 'y_true' [stratify=y_true]
X_train, test_df, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2)
# split the train data into train and cross validation by maintaining same distribution of output variable 'y_train' [stratify=y_train]
train_df, cv_df, y_train, y_cv = train_test_split(X_train, y_train, stratify=y_train, test_size=0.2)
```

Milestone 5: 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 Three classification algorithms KNN, Logistic Regression and Random Forest Classifier. The best model is saved based on its performance.

Link:

<https://colab.research.google.com/drive/15Euimwx7gKKYFRoUQAenxf8fjp2x4rzz?usp=sharing>

Milestone 6: 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.

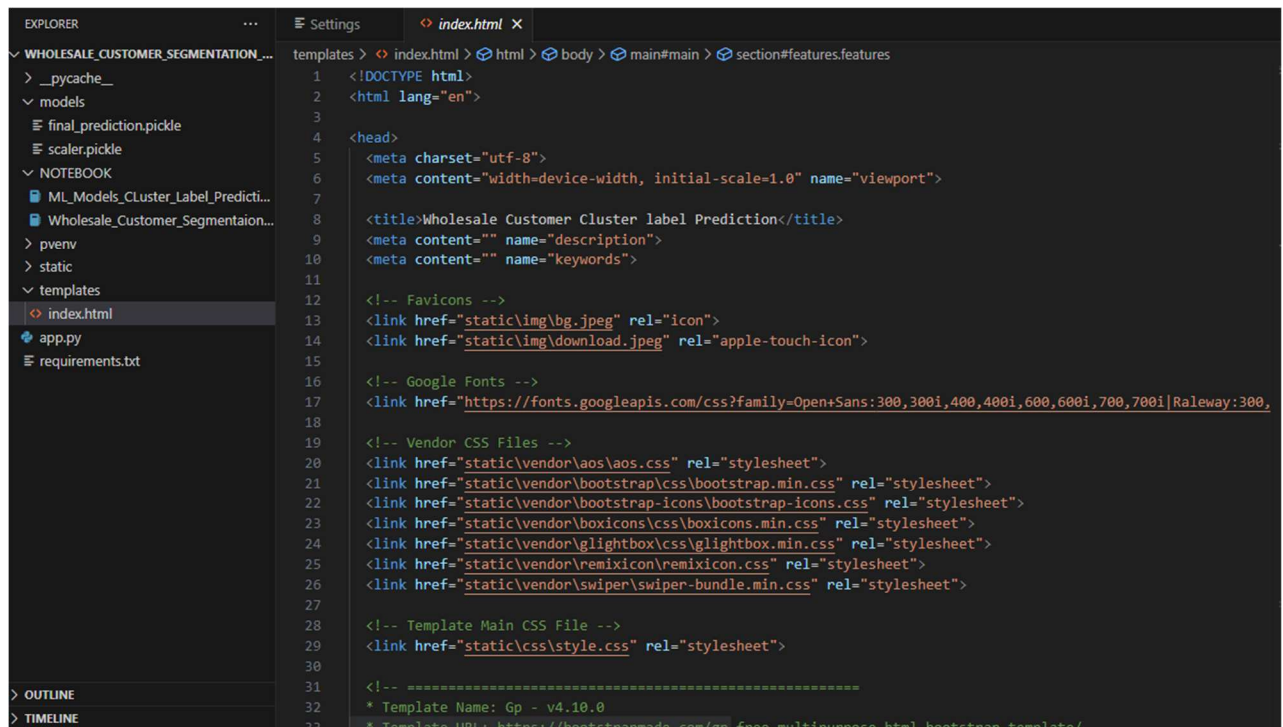
This section has the following tasks

- Building HTML Pages
- Building serverside script

Activity1: Building Html Pages:

For this project create three HTML files namely

- Index.html
and save them in templates folder.



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' panel displays the project structure for 'WHOLESALE_CUSTOMER_SEGMENTATION...'. The 'templates' folder is expanded, showing 'index.html' selected. The main editor area displays the content of 'index.html', which is an HTML template. The code includes a DOCTYPE declaration, HTML lang attribute, a head section with meta tags for charset, viewport, title, description, and keywords, and several link tags for favicons, Google Fonts, and various CSS files (vendor, bootstrap, bootstrap-icons, boxicons, lightbox, remixicon, swiper, and a main style.css). The title is 'Wholesale Customer Cluster label Prediction'. The code is numbered from 1 to 33.

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="utf-8">
6   <meta content="width=device-width, initial-scale=1.0" name="viewport">
7
8   <title>Wholesale Customer Cluster label Prediction</title>
9   <meta content="" name="description">
10  <meta content="" name="keywords">
11
12  <!-- Favicons -->
13  <link href="static\img\bg.jpeg" rel="icon">
14  <link href="static\img\download.jpeg" rel="apple-touch-icon">
15
16  <!-- Google Fonts -->
17  <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Raleway:300,
18
19  <!-- Vendor CSS Files -->
20  <link href="static\vendor\aos\aos.css" rel="stylesheet">
21  <link href="static\vendor\bootstrap\css\bootstrap.min.css" rel="stylesheet">
22  <link href="static\vendor\bootstrap-icons\bootstrap-icons.css" rel="stylesheet">
23  <link href="static\vendor\boxicons\css\boxicons.min.css" rel="stylesheet">
24  <link href="static\vendor\lightbox\css\lightbox.min.css" rel="stylesheet">
25  <link href="static\vendor\remixicon\remixicon.css" rel="stylesheet">
26  <link href="static\vendor\swiper\swiper-bundle.min.css" rel="stylesheet">
27
28  <!-- Template Main CSS File -->
29  <link href="static\css\style.css" rel="stylesheet">
30
31  <!-- =====
32  * Template Name: Gp - v4.10.0
33  * Template URI: https://bootstrapmade.com/gp-free-multipurpose-html-bootstrap-template/
```

Activity 2: Build Python code:

Import the libraries

```
from flask import Flask, render_template, url_for, request
import pickle as p
import pickle
from flask import Flask, request, jsonify, render_template
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

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.

```
modelfile = 'models/final_prediction.pickle'
model = p.load(open(modelfile, 'rb'))
scaler= pickle.load(open('models/scaler.pickle','rb'))
app = Flask(__name__)
```

Render HTML page:

```
@app.route('/')
def welcome():
    return render_template('index.html')
```

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:

```

@app.route('/predict',methods =['GET','POST'])
def predict():
    Channel = float(request.form["Channel"])
    Region =float(request.form['Region'])
    Fresh = float(request.form['Fresh'])
    Milk=float(request.form['Milk'])
    Grocery = float(request.form['Grocery'])
    Frozen = float(request.form['Frozen'])
    Detergents_Paper= float(request.form['Detergents_Paper'])
    Delicassen= float(request.form['Delicassen'])

    total = [[Channel,Region,Fresh,Milk,Grocery,Frozen,Detergents_Paper,Delicassen]]
    prediction = model.predict(scaler.transform(total))
    prediction = int(prediction[0])

    if prediction==0:
        return render_template('index.html',predict="Customer Belongs to Cluster Label 0")

    if prediction==1:
        return render_template('index.html',predict="Customer Belongs to Cluster Label 1")
    if prediction==2:
        return render_template('index.html',predict="Customer Belongs to Cluster Label 2")

    if prediction==3:
        return render_template('index.html',predict="Customer Belongs to Cluster Label 3")
    else:
        return render_template('index.html',predict="Customer Belongs to Cluster Label 4")

```

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 submit.html page earlier.

Main Function:

```

67
68 if __name__ == "__main__":
69     app.run(debug = True)

```

Ln 69, Col 26 UTF-8 CRLF Python 3.10.6 64-bit Go Live

Activity 3: Run the application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict, enter the inputs, click on the submit button, and see the result/prediction on the web.

Final Output :

⌂ ⓘ http://127.0.0.1:5000/predict ⌂ ☆ ❤ ⚙

WHOLESALE CUSTOMER CLUSTER LABEL PREDICTION.

[Home](#) [About](#) [Contact](#) [Predict](#)

Channel

Region

Fresh

Milk

Grocery

Frozen

Detergents_Paper

Delicassen

Wholesale Customer Cluster label Prediction: **Customer
Belongs to Cluster Label 0**