TEAM ID: **592110**

Wholesale Customer Segmentation Analysis using ML

Project Description:

This project focuses on analyzing the spending behavior of wholesale customers with the goal of uncovering patterns and identifying growth opportunities. The dataset contains annual spending data (in monetary units) across various product categories, such as fresh, milk, grocery, frozen, detergents and paper, and delicatessen products. Additionally, it includes information on the customer's channel (hotel/restaurant/cafe or retail) and region (Lisbon, Oporto, or other).

The primary objectives of this project are:

1. **Segmentation of Customers: Identify distinct customer segments based on their spending behaviors. By doing so, the project aims to reveal unique patterns and characteristics within the dataset.

2. **Insight Generation: Provide insights into the identified customer segments. Understanding the specific needs and preferences of each segment can guide wholesale businesses in tailoring their marketing strategies and product offerings.

3. **Predictive Modeling: Develop predictive models to categorize new customers into the identified segments. This can assist businesses in real-time decision-making and customization of services based on predicted spending behaviors.

Potential applications of the project outcomes include:

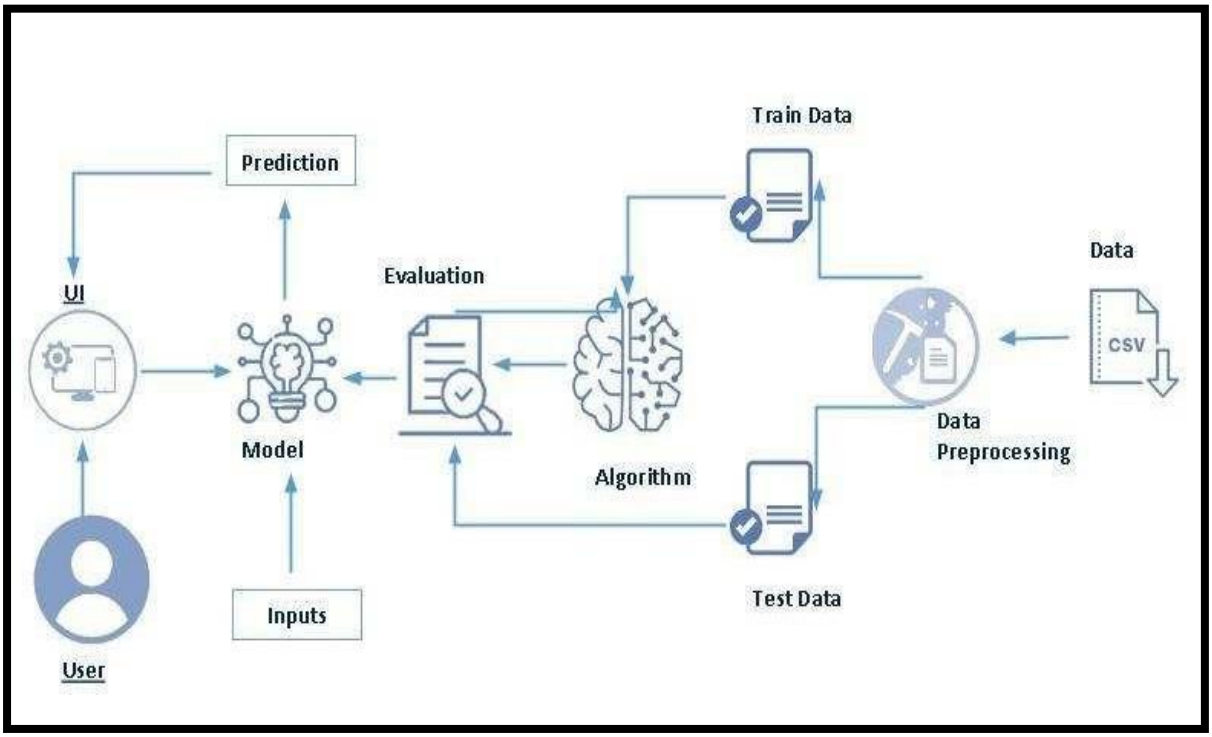
- Targeted Marketing Strategies: Tailor marketing campaigns to specific customer segments, optimizing the effectiveness of promotional efforts.

- Product Offerings: Adjust product offerings based on the preferences of different customer segments, ensuring a more personalized and appealing selection.

- Resource Allocation: Efficiently allocate resources by understanding the varying needs of different customer groups.

By undertaking this analysis, wholesale businesses can gain actionable insights, enhancing their ability to adapt and cater to the diverse spending behaviors of their customers.

Technical Architecture:



Pre requisites:

To complete this project we must require few software's, concepts and packages

- **Anaconda navigator and pycharm:**

We use anaconda for distribution of the Python and R programming languages for data science and machine learning. We utilize it by using pre-installed Data science libraries and for package management.

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:

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

ML concepts

Supervised learning is a machine learning paradigm where the algorithm is trained on a labeled dataset, learning the mapping from input data to corresponding output labels. The model generalizes from the training data to make predictions on new, unseen data. It involves a clear target variable, and the goal is to minimize the difference between predicted and actual outcomes.

Unsupervised learning is a machine learning approach where the algorithm explores patterns and structures in unlabeled data without explicit guidance. It aims to discover inherent relationships or groupings within the data, helping to uncover hidden insights or clusters. Unlike supervised learning, there's no predefined target variable, and the algorithm learns independently from the data's inherent structure.

K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm. It classifies a data point based on the majority class of its k nearest neighbors in the feature space. The choice of k influences the model's sensitivity to local variations, and it's commonly used for classification tasks.

Logistic Regression is a binary classification algorithm used in machine learning. It models the probability of an instance belonging to a particular class, transforming linear combinations of input features into values between 0 and 1 using the logistic function. The decision boundary is set to classify instances based on whether their predicted probability is above or below a specified threshold.

Evaluation metrics assess the performance of machine learning models. Common metrics include accuracy, measuring the proportion of correctly predicted instances; precision, quantifying the accuracy of positive predictions; and recall, gauging the ability to capture all positive instances. These metrics collectively aid in understanding the model's effectiveness and potential trade-offs.

Flask Basics

- Model Integration:
- Input Handling:
- Output Format:
- Endpoint Creation:
- Error Handling:
- Security Considerations:
- Scalability:
- Asynchronous Requests (Optional):
- Logging:
- Documentation:

Project Objectives:

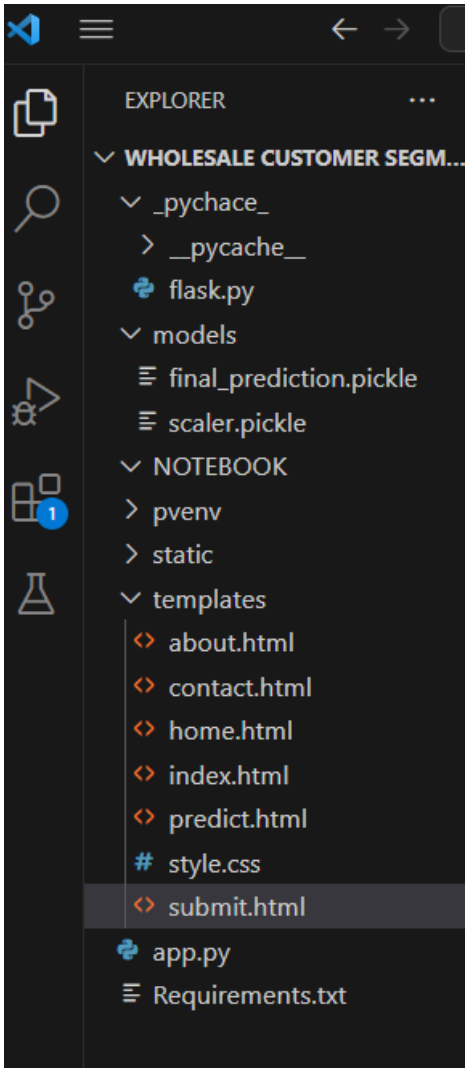
- Know fundamental concepts and techniques used for machine learning.
- Gain a broad understanding about data.
- Have knowledge on pre-processing the data/transformation techniques on outlier and somevisualization concepts.

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:

The Project folder which contains files as shown below:



- We are building a flask application which needs HTML pages stored in the templatesfolder 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 comprehensively analyze the spending behavior of wholesale customers to pinpoint growth opportunities. The dataset includes annual spending data across product categories like fresh, milk, grocery, frozen, detergents and paper, and delicatessen. It also incorporates customer information on channels (hotel/restaurant/café or retail) and regions (Lisbon, Oporto, or other).

Utilizing unsupervised machine learning, particularly clustering algorithms, the project seeks to group customers based on similar spending patterns. The primary goal is to unearth distinct customer segments, offering insights into tailoring marketing strategies and product offerings for each segment. This strategic customization aims to enhance overall customer satisfaction and retention.

Moreover, the project aims to identify growth prospects, uncovering underrepresented products among customers and determining segments receptive to new offerings. In essence, the Wholesale Customer Segmentation project endeavors to equip wholesale businesses with valuable insights to optimize operations and foster growth.

Activity 2: Business Requirements

For Wholesale Customer Segmentation, the project must fulfill the following business requirements:

1. **Accurate Forecasting:**

   - The predictive model should accurately forecast the spending behavior of wholesale customers, providing reliable insights for informed decision-making.

2. **User-Friendly Interface:**

   - The predictor must feature a user-friendly interface for easy navigation and comprehension. Results should be presented clearly to help wholesale businesses optimize operations, enhance customer satisfaction, and improve retention.

Activity 3: Literature Survey

Wholesale customer segmentation involves categorizing customers based on shared characteristics, aiding businesses in understanding and customizing marketing and sales strategies. Existing literature highlights its significance, with studies showing that segmented businesses are more likely to achieve revenue goals and effectively target marketing campaigns, develop customer-centric products, and increase satisfaction and retention.

Common segmentation methods include:

- Geographic Segmentation: Dividing customers by location for localized campaigns.

- Demographic Segmentation: Categorizing based on age, gender, income, etc., for targeted offerings.

- Behavioural Segmentation: Grouping based on buying habits for tailored marketing and product development.

Wholesale customer segmentation is universally valuable, enabling businesses to align strategies with customer needs.

Activity 4: Social or Business Impact

The Wholesale Customer Segmentation project can have significant social and business impacts:

1. Increased Customer Satisfaction and Retention

- Tailoring strategies to customer segments enhances satisfaction and retention, fostering stronger customer-business relationships.

2. Improved Operational Efficiency:

-Identifying growth opportunities streamlines the supply chain, optimizes inventory, and allocates resources efficiently, improving overall operational efficiency.

3. Increased Profitability:

- Enhanced customer satisfaction, improved efficiency, and targeted strategies contribute to increased profitability by generating more revenue, acquiring new customers, and reducing costs.

Milestone 2: Data Collection

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

So thissection 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

Channel

	A	B	C	D	E	F	G	H
1	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents	Delicasse
2	2	3	12669	9656	7561	214	2674	133
3	2	3	7057	9810	9568	1762	3293	177
4	2	3	6353	8808	7684	2405	3516	784
5	1	3	13265	1196	4221	6404	507	178
6	2	3	22615	5410	7198	3915	1777	518
7	2	3	9413	8259	5126	666	1795	145
8	2	3	12126	3199	6975	480	3140	54
9	2	3	7579	4956	9426	1669	3321	256
10	1	3	5963	3648	6192	425	1716	75
11	2	3	6006	11093	18881	1159	7425	209
12	2	3	3366	5403	12974	4400	5977	174
13	2	3	13146	1124	4523	1420	549	49
14	2	3	31714	12319	11757	287	3881	293
15	2	3	21217	6208	14982	3095	6707	60
16	2	3	24653	9465	12091	294	5058	216
17	1	3	10253	1114	3821	397	964	41
18	2	3	1020	8816	12121	134	4508	108
19	1	3	5876	6157	2933	839	370	447
20	2	3	18601	6327	10099	2205	2767	318
21	1	3	7780	2495	9464	669	2518	50
22	2	3	17546	4519	4602	1066	2259	212
23	1	3	5567	871	2010	3383	375	56
24	1	3	31276	1917	4469	9408	2381	433
25	2	3	26373	36423	22019	5154	4337	1652
26	2	3	22647	9776	13792	2915	4482	577

Milestone 3: Visualizing and analysing the data

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

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

Activity 1: Importing the libraries

```
# Import all required Libraries:

import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from imblearn.over_sampling import SMOTE
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")
import six
import sys
sys.modules['sklearn.externals.six'] = six
from mlxtend.classifier import StackingClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
```

Activity 2: Read the Dataset

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

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

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
warnings.filterwarnings("ignore")
import six
import sys
sys.modules['sklearn.externals.six'] = six
from mlxtend.classifier import StackingClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
```

```
[ ] df = pd.read_csv("/content/drive/MyDrive/DATASETS/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 havedisplayed two different graphs such as distplot and countplot.

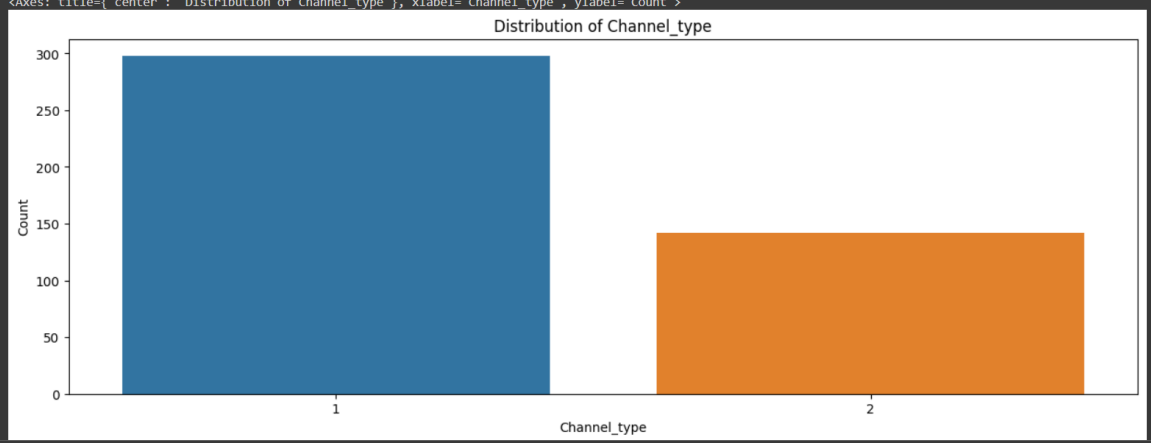
- Seaborn package provides a wonderful function distplot. With the help of distplot, we canfind the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
#Univariate Analysis
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

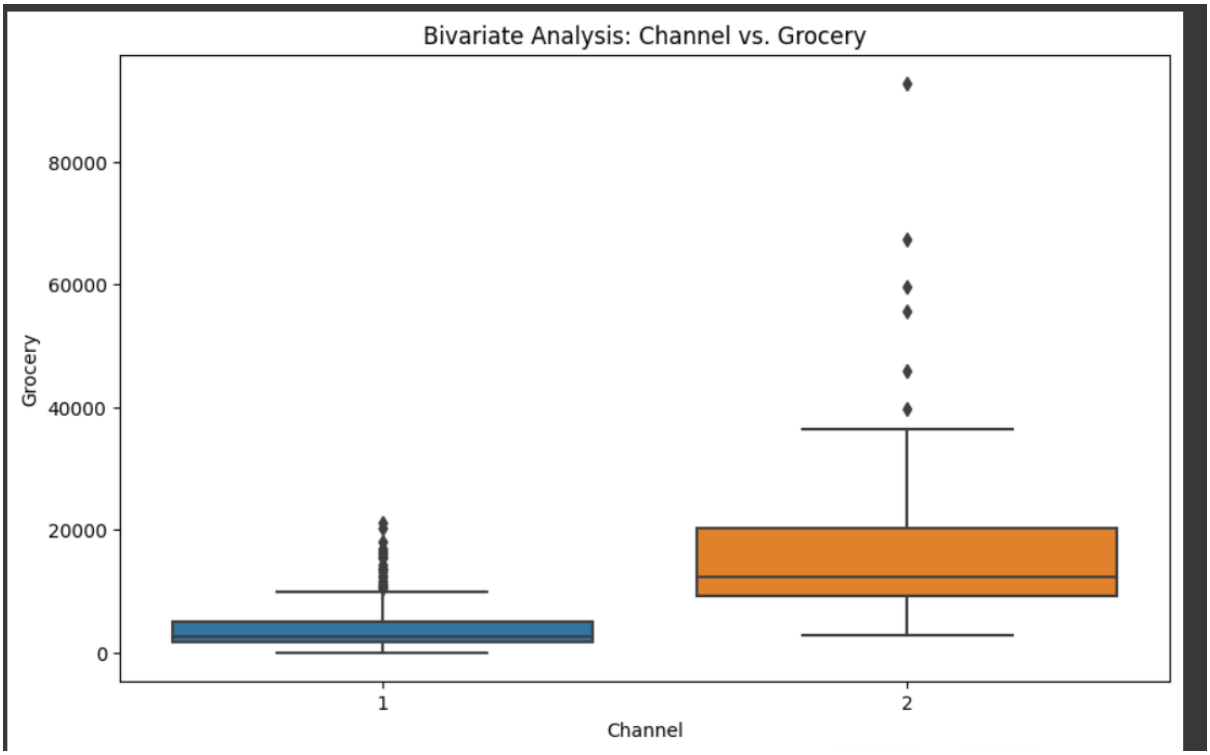
```
plt.figure(figsize=(15,5))
plt.title('Distribution of Channel_type')
sns.barplot(x='Channel_type', y='Count', data = Channel_df)
```

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



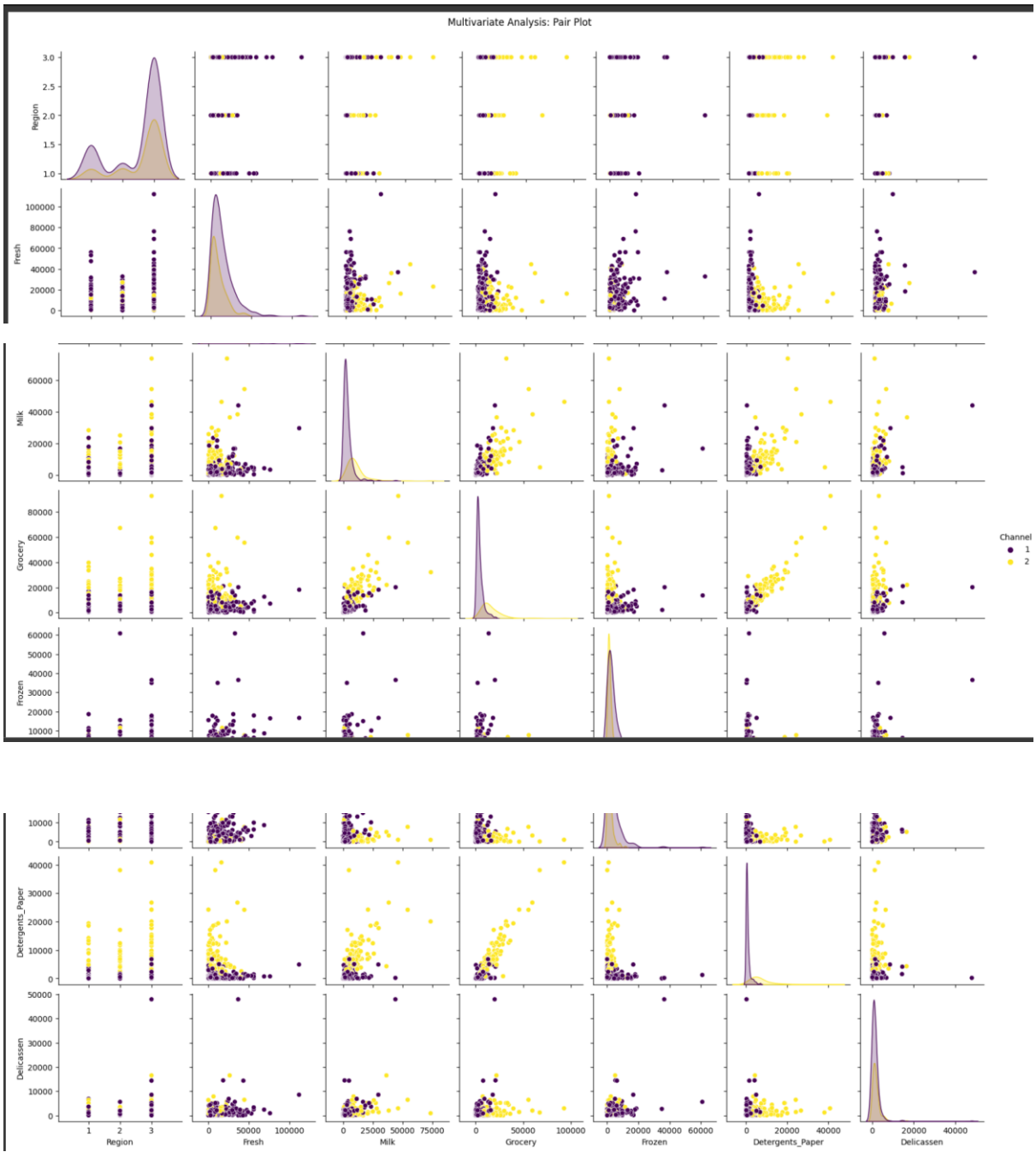
Activity 4: Bivariate analysis

```
# Bivariate Analysis
plt.figure(figsize=(10, 6))
sns.boxplot(x='Channel', y='Grocery', data=df)
plt.title('Bivariate Analysis: Channel vs. Grocery')
plt.xlabel('Channel')
plt.ylabel('Grocery')
plt.show()
```



Activity 5: Multivariate analysis

```
# Multivariate Analysis: Pair Plot
sns.pairplot(df, hue='Channel', palette='viridis', diag_kind='kde')
plt.suptitle('Multivariate Analysis: Pair Plot', y=1.02)
plt.show()
```



Activity 6: Descriptive analysis

```
#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

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 somuch of randomness so we need to clean the dataset properly in order to fetch good results. Thisactivity 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

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() is used.

To sum those null values we use .sum() function in it.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Channel           440 non-null    int64
 1   Region            440 non-null    int64
 2   Fresh             440 non-null    int64
 3   Milk              440 non-null    int64
 4   Grocery           440 non-null    int64
 5   Frozen            440 non-null    int64
 6   Detergents_Paper  440 non-null    int64
 7   Delicassen        440 non-null    int64
dtypes: int64(8)
memory usage: 27.6 KB
```

```
[ ] #checking for null values
    df.isnull().sum()
```

```
Channel            0
Region             0
Fresh              0
Milk               0
Grocery            0
Frozen             0
Detergents_Paper   0
Delicassen         0
dtype: int64
```

Activity 2: Handling outliers

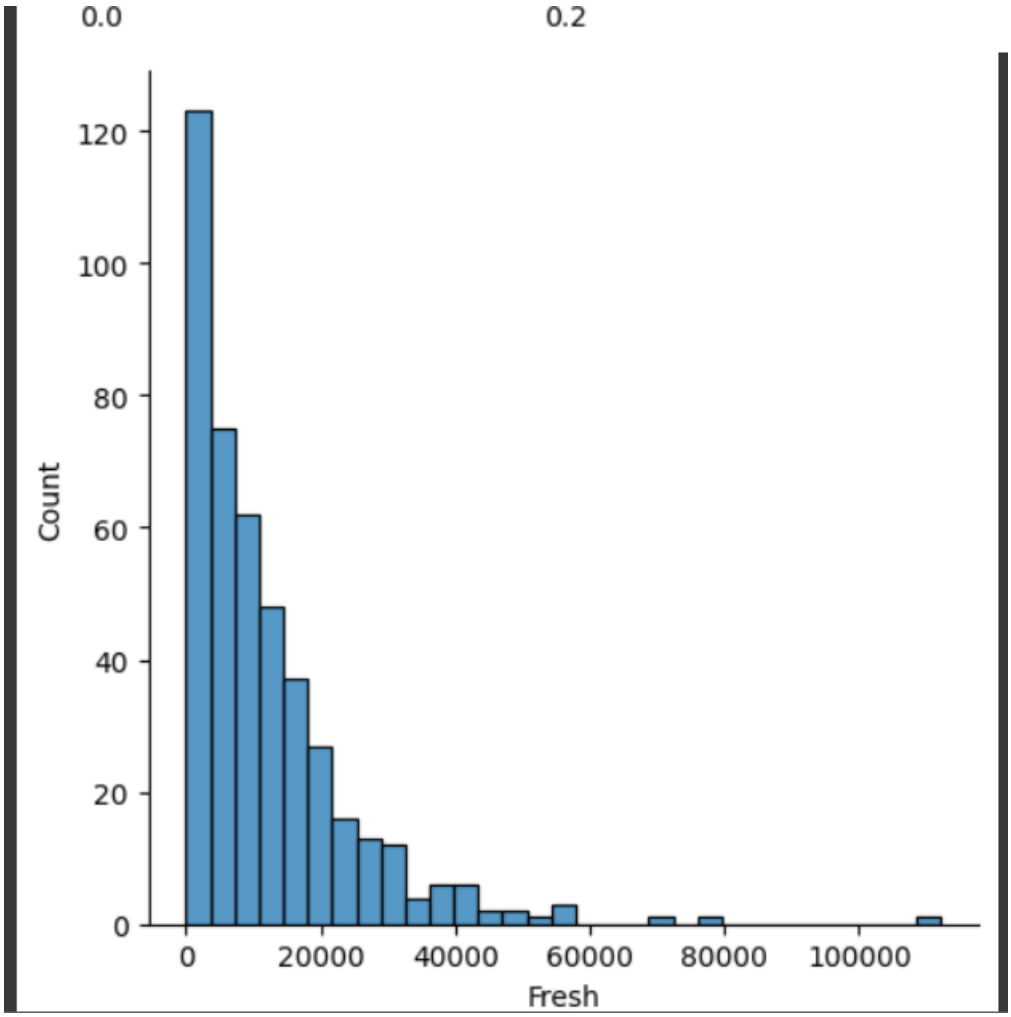
With the help distribution, outliers are visualized. And here we are going to find upper bound andlower bound of all features with some mathematical formula.

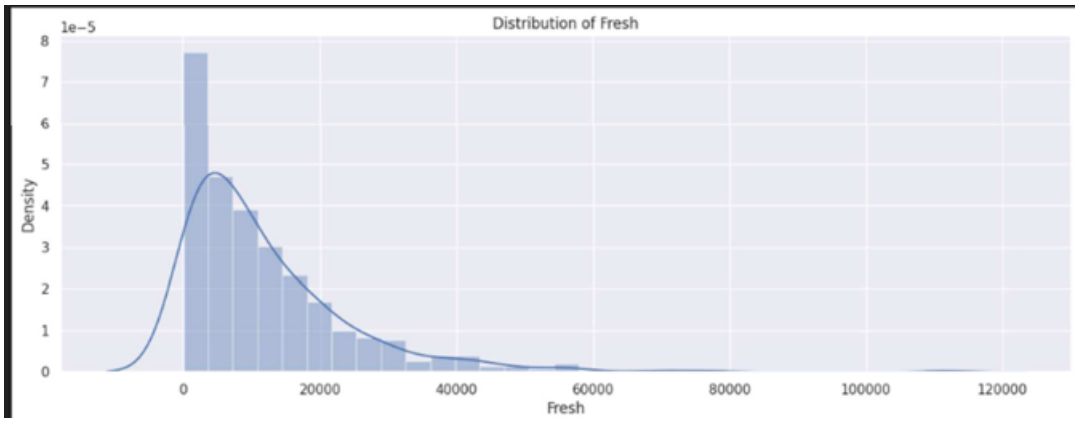
- From the below diagram, we could visualize that Distribution of feature has outliers orNot.

```
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
```

```
#Distribution of FRESH
plt.figure(figsize=(15,5))
plt.title("Distribution of FRESH")
sns.displot(df['Fresh'])
```





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 splitthe data set

```
# split the data into test and train by maintaining same distribution of output varaible '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 varaible '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

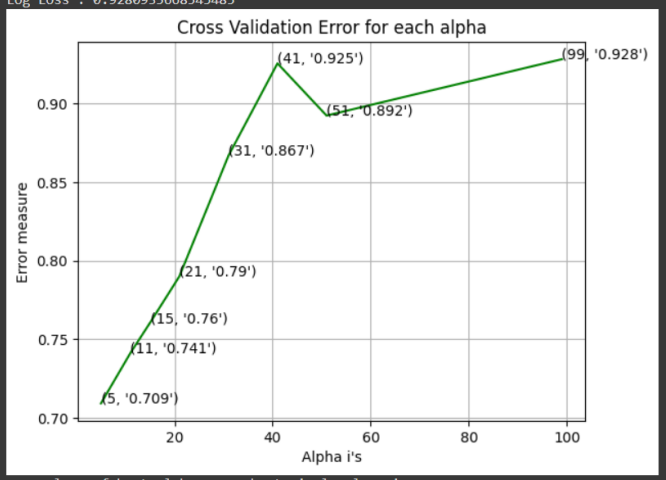
Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying Three classification algorithms **KNN, Logistic Regression and Random Forest Classifier**. The best model is saved based on its performance.

K Nearest Neighbour Classification

Hyper parameter tuning

```
alpha = [5, 11, 15, 21, 31, 41, 51, 99]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = KNeighborsClassifier(n_neighbors=i)
    clf.fit(train_df, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df, y_train)
    sig_clf_probs = sig_clf.predict_proba(cv_df)
    cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilites we use log-probability estimates
    print("Log Loss :",log_loss(y_cv, sig_clf_probs))
```

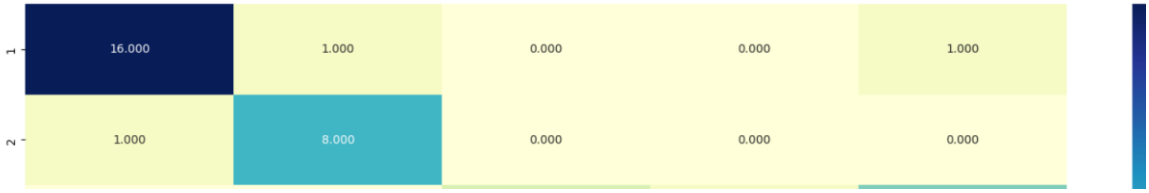
```
Log Loss : 0.8922531184983462
for alpha = 99
Log Loss : 0.9280935668545485
```




```
For values of best alpha =  5 The train log loss is: 0.6062594313775383
For values of best alpha =  5 The cross validation log loss is: 0.7092796933538674
For values of best alpha =  5 The test log loss is: 0.7691966283741848
```

```
clf = KNeighborsClassifier(n_neighbors=alpha[best_alpha])
predict_and_plot_confusion_matrix(train_df.values, y_train.values, cv_df.values, y_cv.values, clf)
```

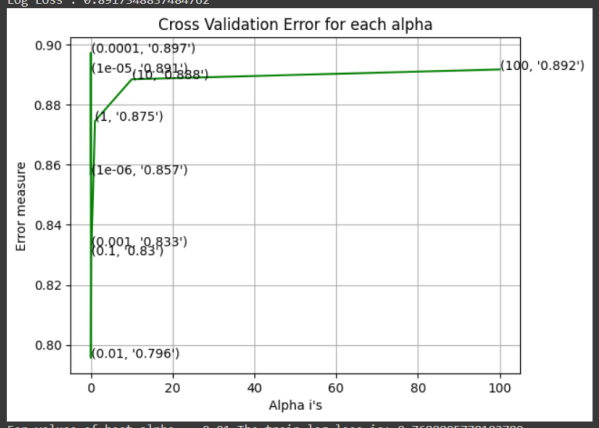
```
Log loss : 0.7092796933538674
Number of mis-classified points : 55.25352112676056
-------------------- Confusion matrix --------------------
```



LOGISTIC REGRESSION :

```
#Logistic Regression
#Hyper paramter tuning
alpha = [10 ** x for x in range(-6, 3)]
cv_log_error_array = []
for i in alpha:
    print("for alpha =", i)
    clf = SGDClassifier(class_weight='balanced', alpha=i, penalty='l2', loss='log', random_state=42)
    clf.fit(train_df,y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(train_df, y_train)
    sig_clf_probs = sig_clf.predict_proba(cv_df)
    cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    # to avoid rounding error while multiplying probabilites we use log-probability estimates
    print("Log Loss :",log_loss(y_cv, sig_clf_probs))
```

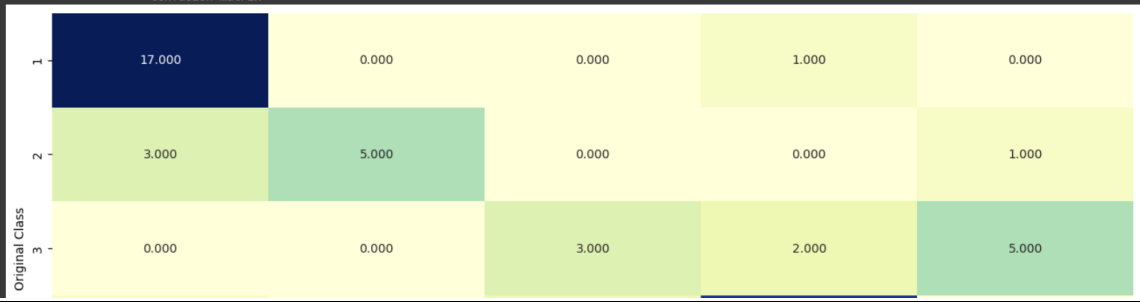
```
Log Loss : 0.8884711594170405
for alpha = 100
Log Loss : 0.8917348837484762
```




```
For values of best alpha =  0.01 The train log loss is: 0.7688805778182789
For values of best alpha =  0.01 The cross validation log loss is: 0.7957154162344837
For values of best alpha =  0.01 The test log loss is: 0.7703275773460941
```

```
clf = SGDClassifier(class_weight='balanced', alpha=alpha[best_alpha], penalty='l2', loss='log', random_state=42)
predict_and_plot_confusion_matrix(train_df.values, y_train.values, cv_df.values, y_cv.values, clf)
```

```
Log loss : 0.7957154162344836
Number of mis-classified points : 55.0
-------------------- Confusion matrix --------------------
```



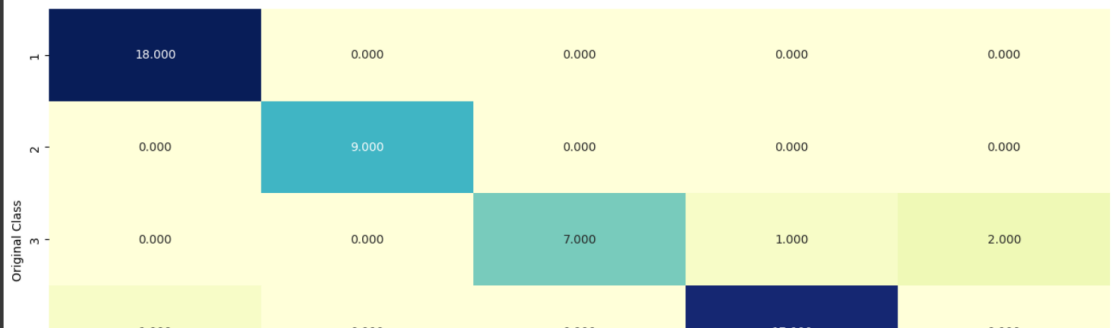
RANDOM FOREST CLASSIFIER :

```
#Random forest
alpha = [100,200,500,1000,2000]
max_depth = [5, 10]
cv_log_error_array = []
for i in alpha:
    for j in max_depth:
        print("for n_estimators =", i,"and max depth = ", j)
        clf = RandomForestClassifier(n_estimators=i, criterion='gini', max_depth=j, random_state=42, n_jobs=-1)
        clf.fit(train_df,y_train)
        sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
        sig_clf.fit(train_df,y_train)
        sig_clf_probs = sig_clf.predict_proba(cv_df)
        cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
        print("Log Loss :",log_loss(y_cv, sig_clf_probs))
```

```
for n_estimators = 500 and max depth =  10
Log Loss : 0.33627466397253847
for n_estimators = 1000 and max depth =  5
Log Loss : 0.35958773898185126
for n_estimators = 1000 and max depth =  10
Log Loss : 0.3321872393872705
for n_estimators = 2000 and max depth =  5
Log Loss : 0.3613571962299473
for n_estimators = 2000 and max depth =  10
Log Loss : 0.3348180412449007
For values of best estimator =  1000 The train log loss is: 0.14729812788957383
For values of best estimator =  1000 The cross validation log loss is: 0.3321872393872705
For values of best estimator =  1000 The test log loss is: 0.3429767213548854
```

```
clf = RandomForestClassifier(n_estimators=alpha[int(best_alpha/2)], criterion='gini', max_depth=max_depth[int(best_alpha%2)], random_state=42, n_jobs=-1)
# predict_and_plot_confusion_matrix(train_x_onehotCoding, train_y,cv_x_onehotCoding,cv_y, clf)
predict_and_plot_confusion_matrix(train_df.values, y_train.values, cv_df.values, y_cv.values, clf)
```

```
Log loss : 0.3321872393872705
Number of mis-classified points : 55.40845070422535
-------------------- Confusion matrix --------------------
```



We applied 3 different classification algorithms

KNN -55.25 classified points

Logistic Regression – 55.0 classified points

Random Forest Classifier – 55.40 classified points

So we can predict that using RANDOM FOREST CLASSIFIER is best among all the other classification algorithms.

Milestone 6: Application Building

In this section, we will be building a web application that is integrated to the model we built. A UIis provided for the uses where he has to enter the values for predictions. The enter values are givento 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 created six HTML files namely

- **Index.html**
- **Home.html**
- **About.html**
- **Comtact.html**
- **Predict.html**
- **Submit.html**

We create all the files using the above names and save them all

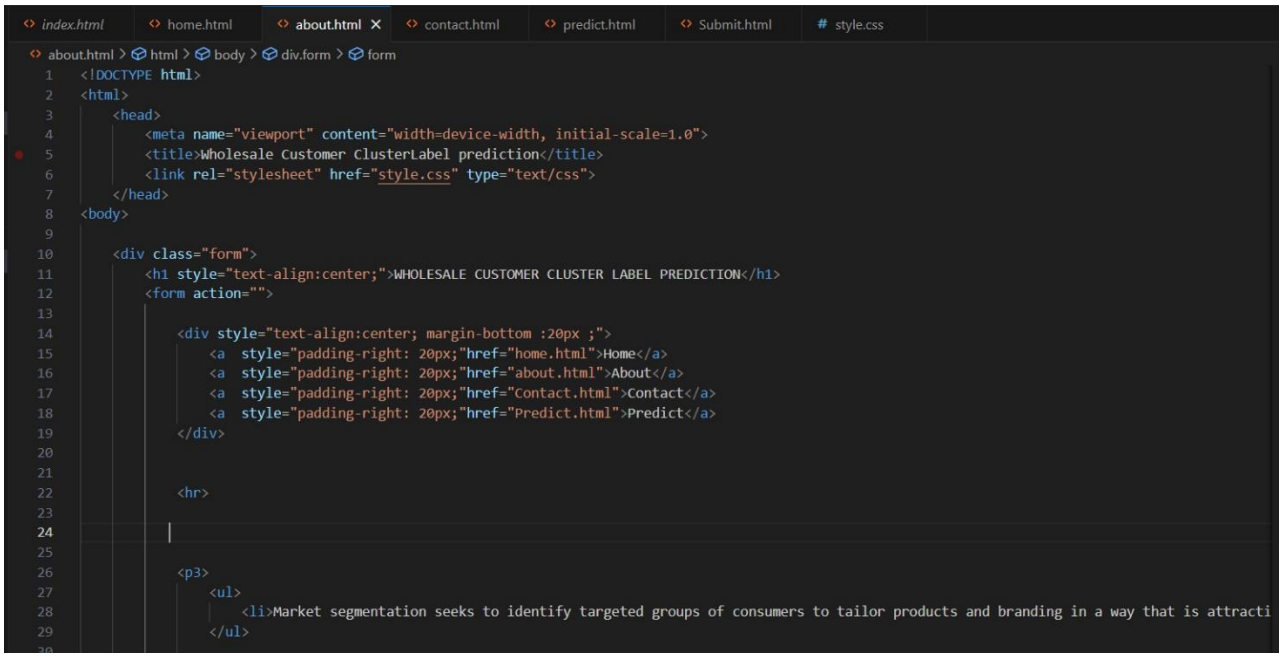
- **Index.html**

```
w  Go  Run  Terminal  Help        customer-segmentation.html webpage

index.html ×  home.html  about.html  contact.html  predict.html  Submit.html  style.css

index.html > html > body > div.form > form

<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Wholesale Customer ClusterLabel prediction</title>
        <link rel="stylesheet" href="style.css" type="text/css">
    </head>
<body>

    <div class="form">
        <h1 style="text-align:center;">WHOLESALE CUSTOMER CLUSTER LABEL PREDICTION</h1>
        <form action="">

            <div style="text-align:center; margin-bottom :20px ;">
                <a  style="padding-right: 20px;"href="home.html">Home</a>
                <a  style="padding-right: 20px;"href="about.html">About</a>
                <a  style="padding-right: 20px;"href="Contact.html">Contact</a>
                <a  style="padding-right: 20px;"href="Predict.html">Predict</a>
            </div>


            <hr>

            <label>Channel &rarr; </label>
            <input type="text" placeholder="Enter The Value"> <br>

            <label>Region  &rarr; </label>
            <input type="text" placeholder="Enter The Value"> <br>
```

- **Home.html**

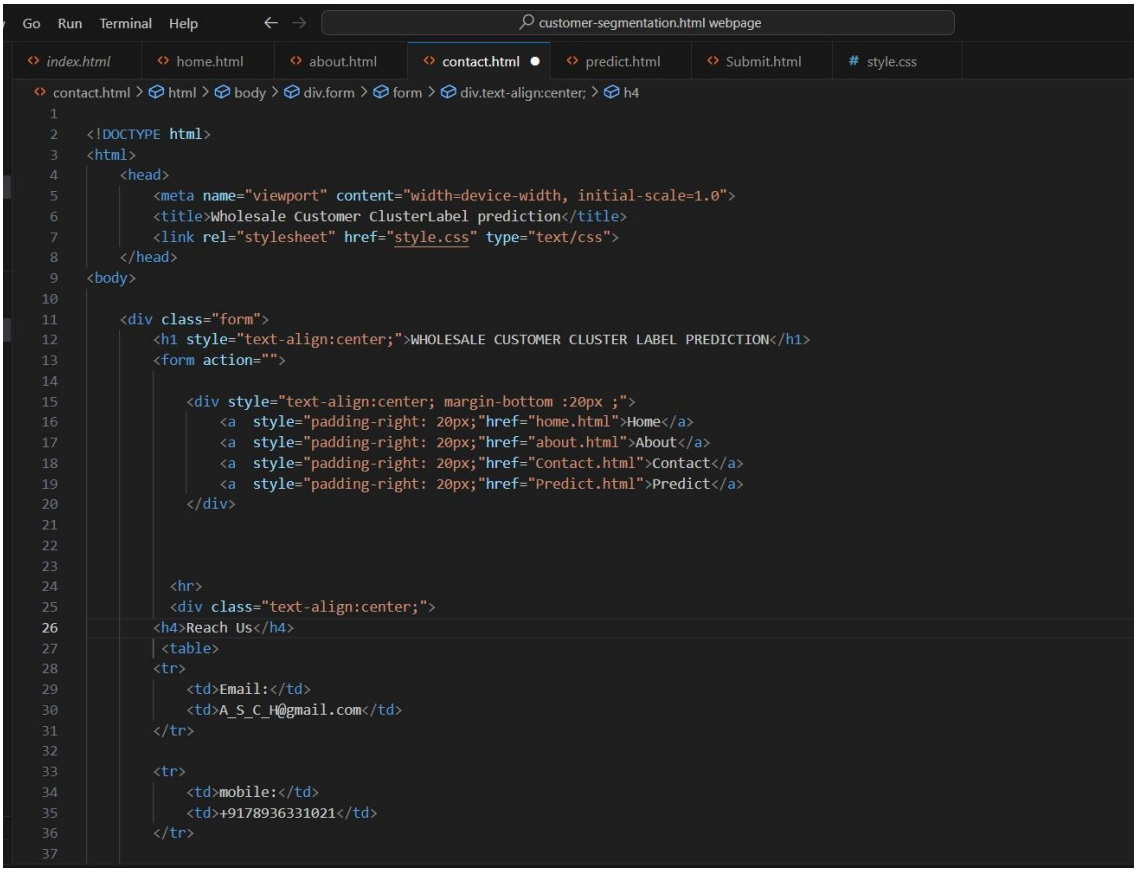
```
index.html  home.html ×  about.html  contact.html  predict.html  Submit.html  style.css

home.html > html > body > div.form > form

<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Wholesale Customer ClusterLabel prediction</title>
        <link rel="stylesheet" href="style.css" type="text/css">
    </head>
<body>

    <div class="form">
        <h1 style="text-align:center;">WHOLESALE CUSTOMER CLUSTER LABEL PREDICTION</h1>
        <form action="">

            <div style="text-align:center; margin-bottom :20px ;">
                <a  style="padding-right: 20px;"href="home.html">Home</a>
                <a  style="padding-right: 20px;"href="about.html">About</a>
                <a  style="padding-right: 20px;"href="Contact.html">Contact</a>
                <a  style="padding-right: 20px;"href="Predict.html">Predict</a>
            </div>


            <hr>

            <label>Channel &rarr; </label>
            <input type="text" placeholder="Enter The Value"> <br>

            <label>Region  &rarr; </label>
            <input type="text" placeholder="Enter The Value"> <br>
```

- **About.html**



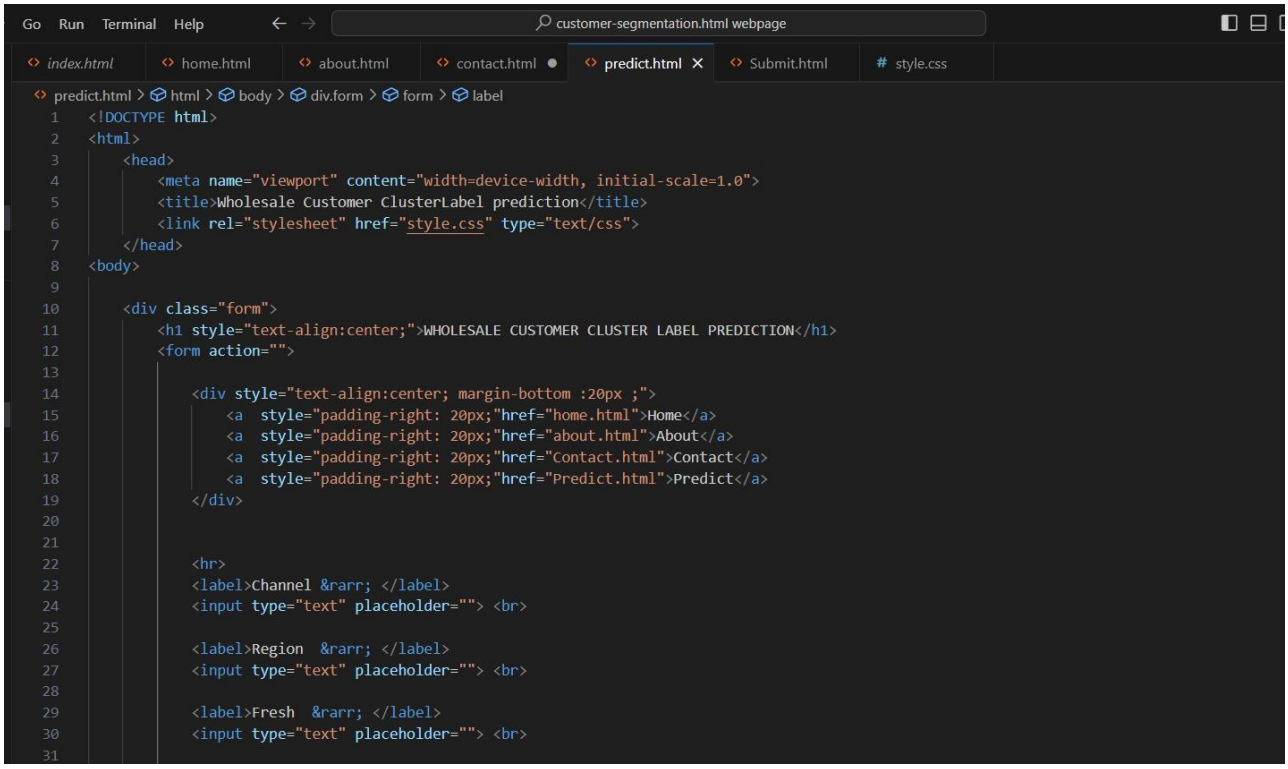
```
index.html  home.html  about.html  contact.html  predict.html  Submit.html  style.css
about.html > html > body > div.form > form
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Wholesale Customer ClusterLabel prediction</title>
        <link rel="stylesheet" href="style.css" type="text/css">
    </head>
<body>

    <div class="form">
        <h1 style="text-align:center;">WHOLESALE CUSTOMER CLUSTER LABEL PREDICTION</h1>
        <form action="">

            <div style="text-align:center; margin-bottom :20px ;">
                <a  style="padding-right: 20px;"href="home.html">Home</a>
                <a  style="padding-right: 20px;"href="about.html">About</a>
                <a  style="padding-right: 20px;"href="Contact.html">Contact</a>
                <a  style="padding-right: 20px;"href="Predict.html">Predict</a>
            </div>


            <hr>


            <p3>
                <ul>
                    <li>Market segmentation seeks to identify targeted groups of consumers to tailor products and branding in a way that is attracti
                </ul>
```

- **Contatct.html**

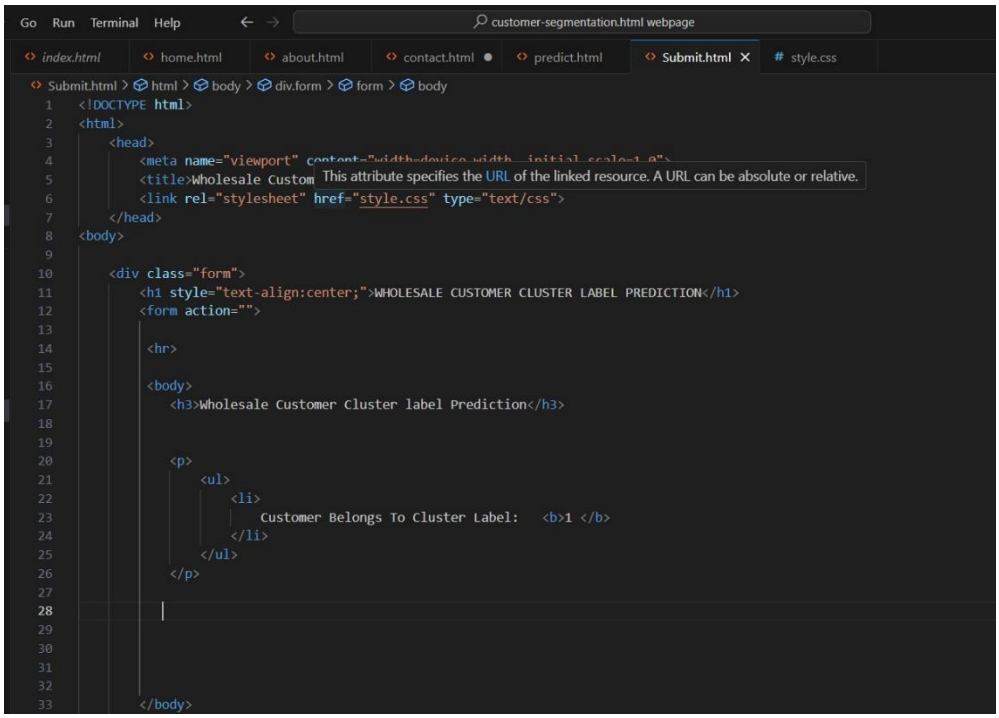


```
Go  Run  Terminal  Help   customer-segmentation.html webpage
index.html  home.html  about.html  contact.html  predict.html  Submit.html  style.css
contact.html > html > body > div.form > form > div.text-align:center; > h4
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Wholesale Customer ClusterLabel prediction</title>
        <link rel="stylesheet" href="style.css" type="text/css">
    </head>
<body>

    <div class="form">
        <h1 style="text-align:center;">WHOLESALE CUSTOMER CLUSTER LABEL PREDICTION</h1>
        <form action="">

            <div style="text-align:center; margin-bottom :20px ;">
                <a  style="padding-right: 20px;"href="home.html">Home</a>
                <a  style="padding-right: 20px;"href="about.html">About</a>
                <a  style="padding-right: 20px;"href="Contact.html">Contact</a>
                <a  style="padding-right: 20px;"href="Predict.html">Predict</a>
            </div>



          <hr>
          <div class="text-align:center;">
        <h4>Reach Us</h4>
        <table>
        <tr>
            <td>Email:</td>
            <td>A_S_C_H@gmail.com</td>
        </tr>

        <tr>
            <td>mobile:</td>
            <td>+9178936331021</td>
        </tr>
```

- **Predict.html**

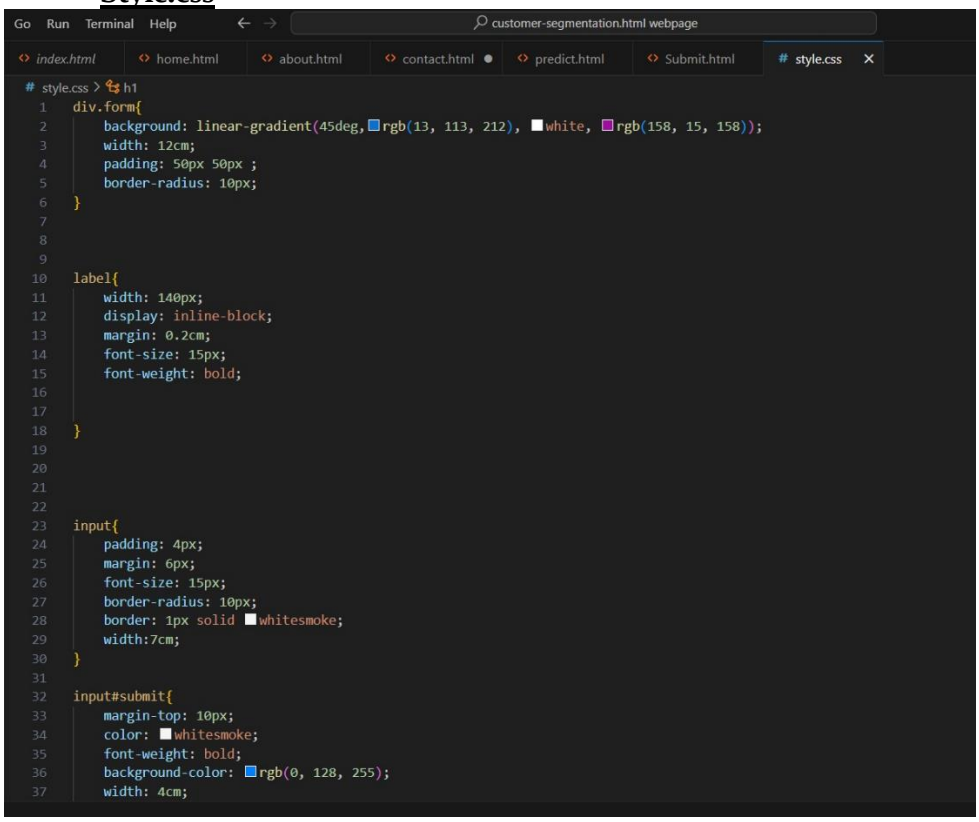


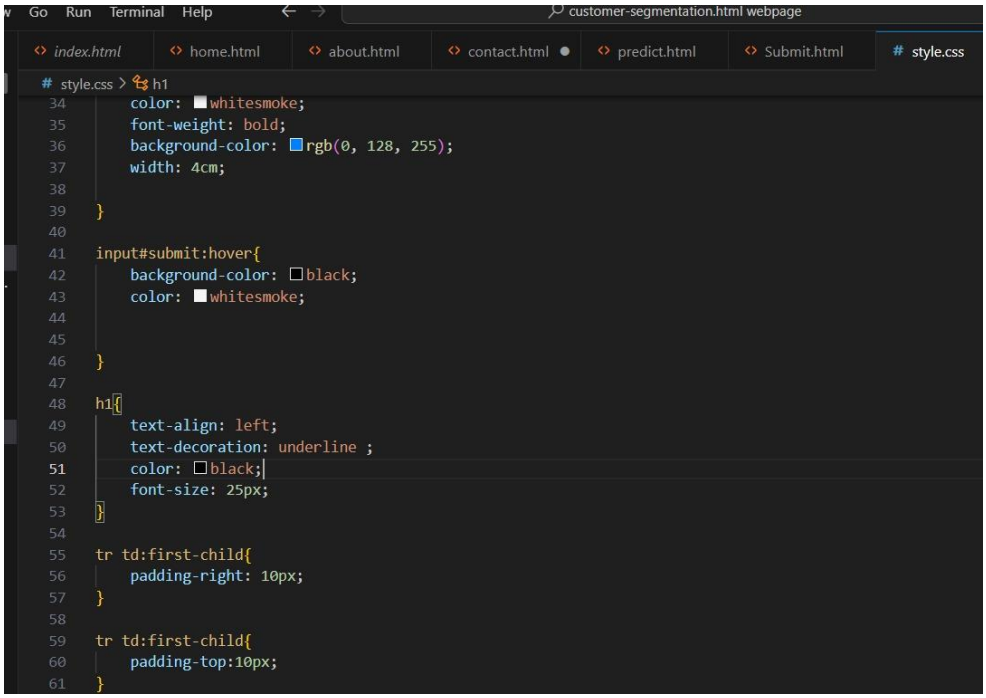
- **Submit.html**



We used CSS to for styling our web application

- **Style.css**





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
```

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 createdearlier.

In the above example, '/' URL is bound with home.html function. Hence, when the home page ofthe 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:

```python
@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 theHTML 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 submit.html page earlier.

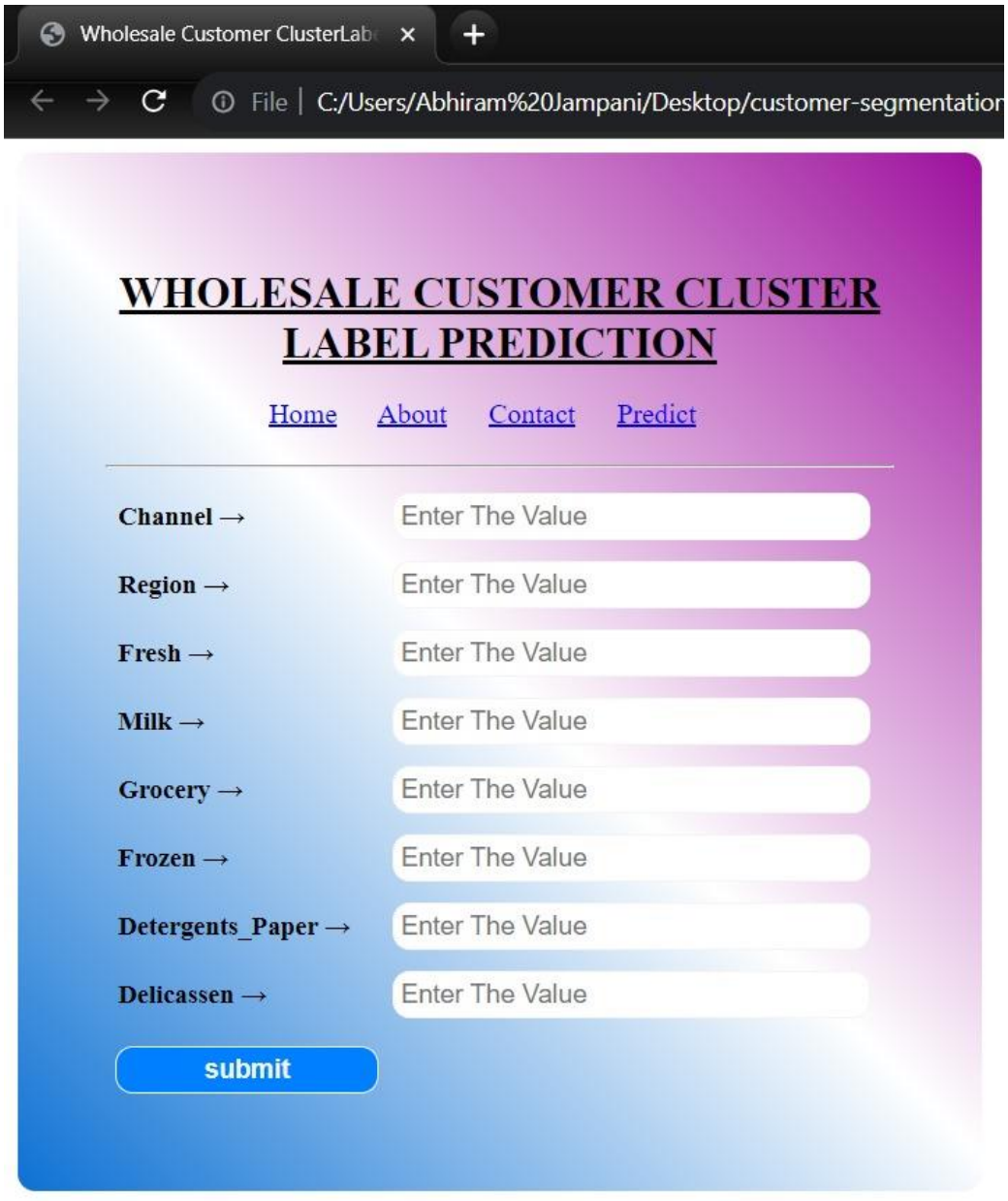
Main Function:

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

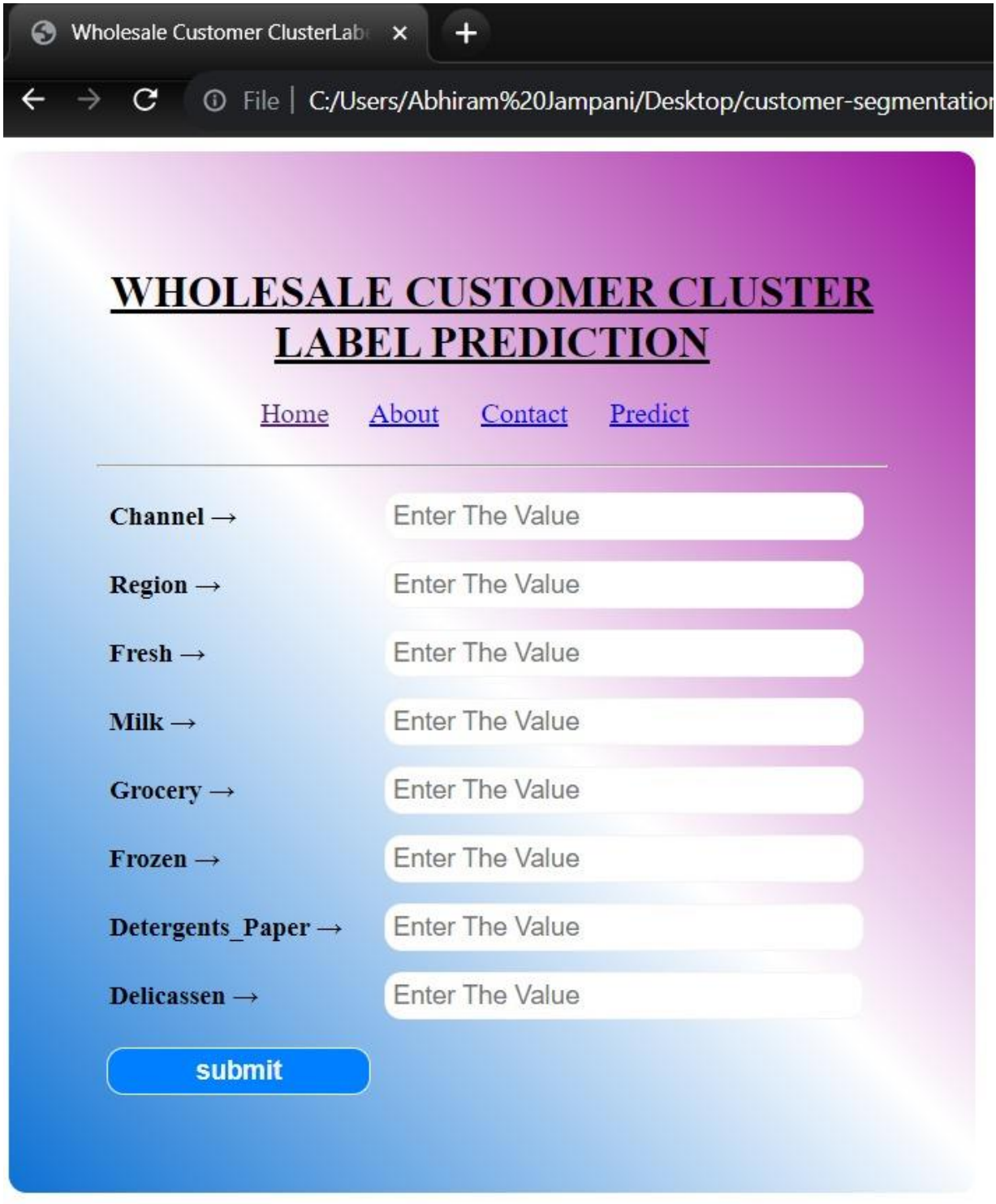
Activity 3: Run the application

- 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 theresult/prediction on the web.

Final Output :

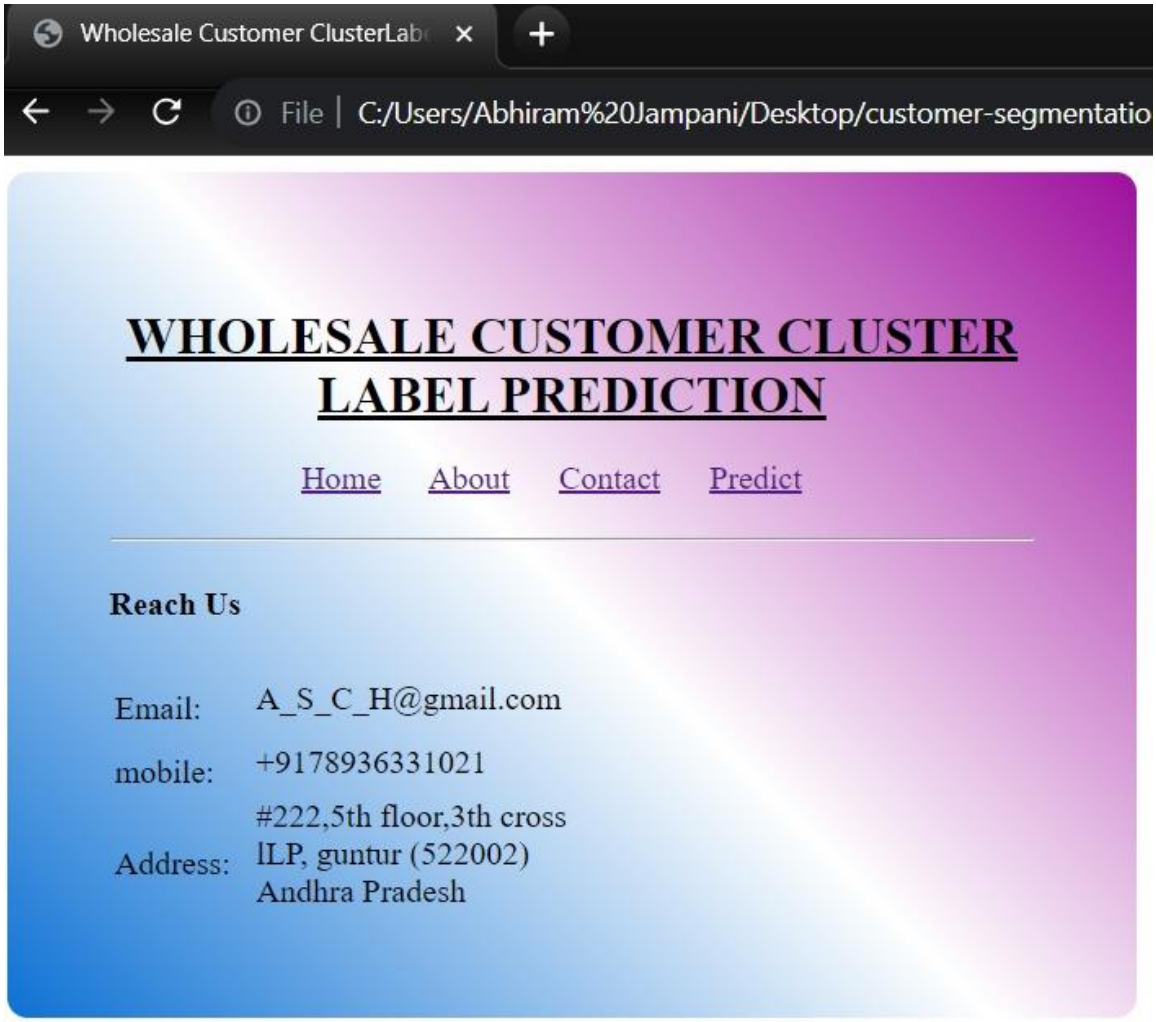


When we select the **HOME** we get:



When we select **ABOUT** we get :

When we select **CONTACT** we get :



When we enter the values to find the prediction for which cluster we get :

AFTER SELECTING THE SUBMIT BUTTON AFTER YOUR VALUES ARE ENTERED WE GET