University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING

Electrical & Computer Engineering and Computer Science

Distributed & Scalable Data Engineering (DSCI-6007)

TECHNICAL REPORT

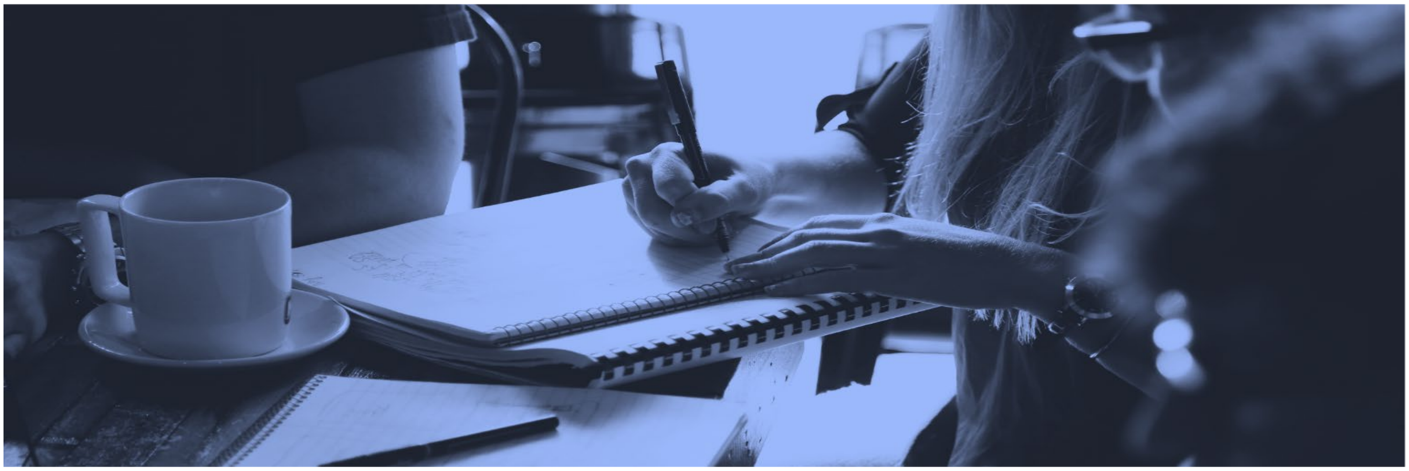
Fall 22

CONTENTS

Customer-Segmentation

EXECUTIVE SUMMARY

- Each Company's goal is to enhance its profits and expand its business, it can be possible only with the help of its loyal customers and by understanding its target customer preferences. In this new era of technology, customers, day-to-day preferences might change.
- Identifying target customers according to their product is not an easy task and creating or inventing new products or technology for a business based on Customer's needs in this day-to-day updating generation. Hence, Marketers must need to know their customer's preferences.



Team Members:

Dinesh Kumar Reddy Desireddygari
Team Leader, Application developer

Pranay Kumar Ganji
Data Scientist

Sai Kiran Gattu
Data Modeling

Satya Shalini Gorrela
Data visualization

HIGHLIGHTS OF THE PROJECT

- In this project, K-means clustering is used to divide the customer data into five different categories based on the features available in the data.
- Each time we use the application the input data used for prediction is stored in a database which allows the model to train irrespective of the timeline.
- Whole application is built in EC2 instance, this allows us to access our application remotely.

ABSTRACT

- In order to have a better Marketer-Customer relationship, every organization needs to perform Customer Segmentation.
- Customer segmentation has the potency to let marketers achieve the goal of understanding their customer's preferences or needs effectively and it also helps to earn a huge market share, associating with the target Customer's group on a priority basis and approaching their customers through potential channels.
- Hence better Customer Segmentation can lead to a better Marketer-Customer relationship, which is the main criterion for marketers present out there.

Video link is attached below- Youtube presentation

GitHub link is attached below – **Git Link**

METHODOLOGY

We imported the data sets into jupyter notebook and performed the analysis. Here, we will import the data and perform data cleaning suitable for modeling and the models are trained using the data. An application will be built to maintain the trained machine learning model and the application will be deployed into the web using the flask server and ec2 enables users to access the application remotely. In this project, we have used CRISP-DM methodology.

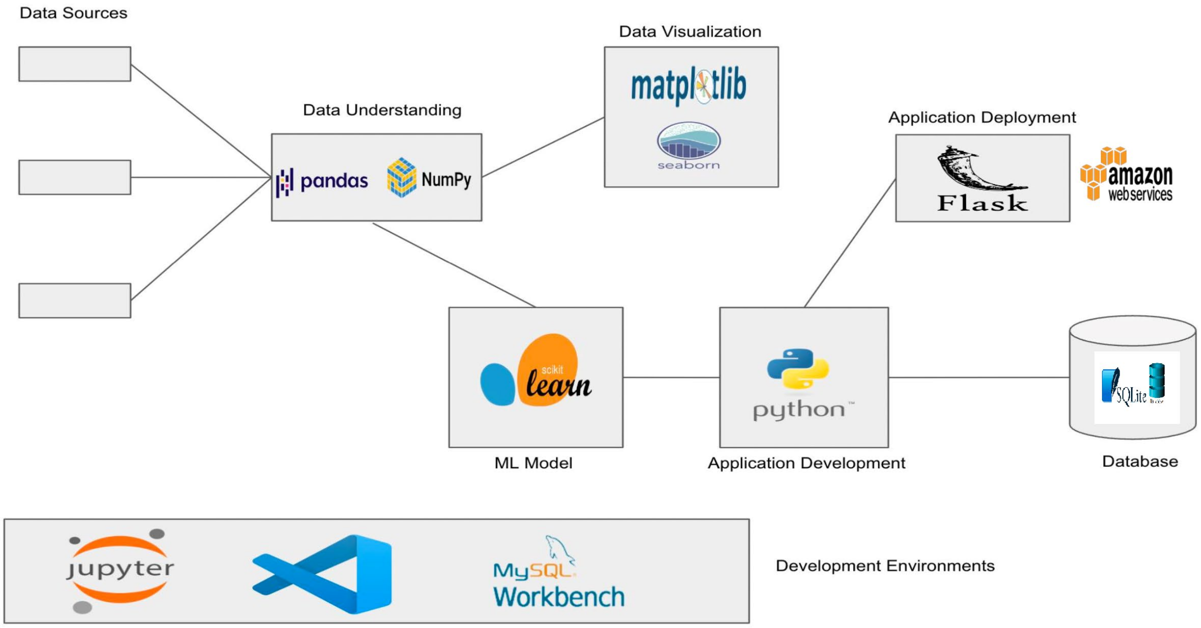
BUSINESS UNDERSTANDING

The process of finding groups (or segments) of a company's consumers that are comparable in terms of one or more certain qualities or features is known as customer segmentation, also known as market segmentation. This classification aims to maximize the value of each customer to your organization by optimizing marketing to each category and ensuring that individual customers receive the most pertinent and appropriate communications.

There are an almost infinite number of potential traits or variables that can be used to categorize clients, but the most prevalent (and readily available) ones are as follows:

- Age, stage of life (retired, new parents, students, etc.), gender, and marital status are examples of personal traits.
- Location, as well as urban/suburban/rural areas, are geographical considerations.
- Purchasing habits, including previous purchases (value, frequency, type of products purchased)

ARCHITECTURE

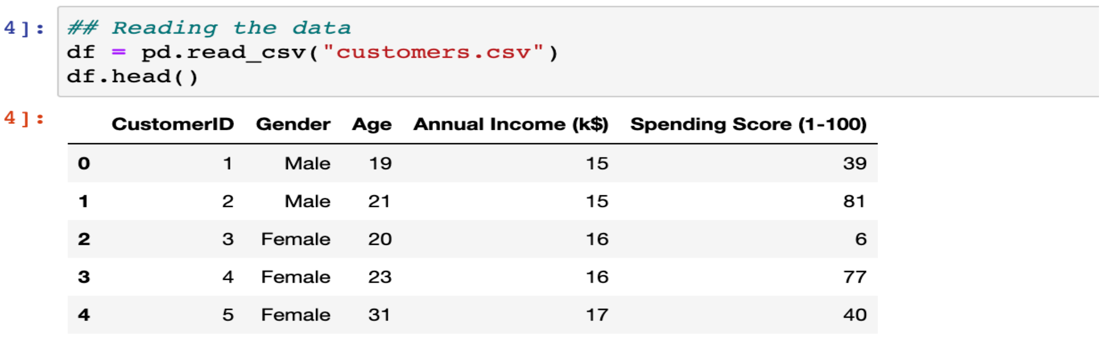


PROJECT REQUIREMENTS

- **Python libraries:**
  Pandas, and NumPy are used for Data Cleaning, Data Wrangling.
  Matplotlib, and Seaborn are used for Data Visualization.
  Sklearn is used for Machine learning Models.
- **Programming languages :**
  Python, HTML are used for Application Development.
- **Database:**
  Sqlite is used for performing CRUD operations on the given data and used for data storage.
- **FLASK:**
  The Flask library is used for the application deployment of the developed application.
- **Development Environments:**
  Visual Studio Code, Jupyter Notebook.

DATA UNDERSTANDING

For this project, we will be used the customer data set from Kaggle. The data consists of 5 columns and 200 rows, all columns are integers except the gender column.



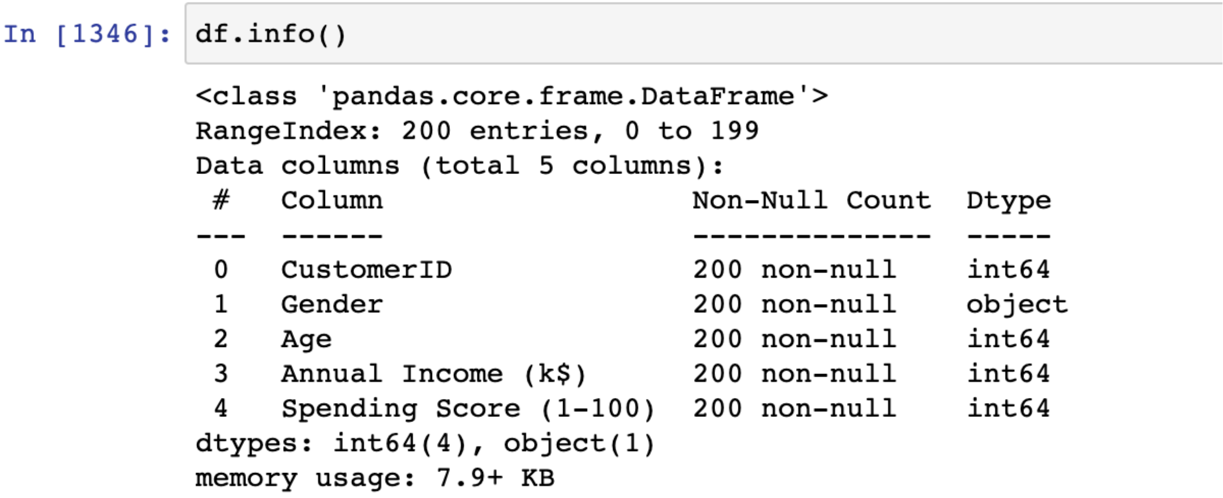
```
4]: ## Reading the data
    df = pd.read_csv("customers.csv")
    df.head()
```

4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

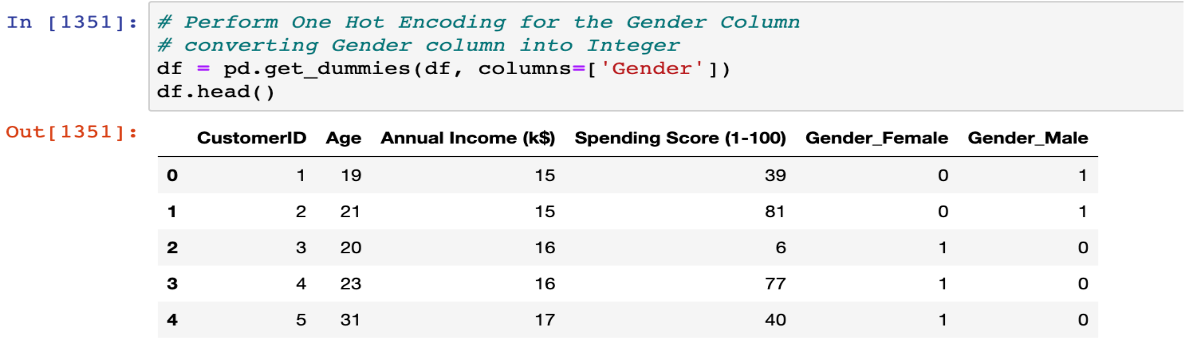
DATA PREPARATION

Using pandas library, the data is made suitable for modeling by dropping the unnecessary columns and filling the null values. Here, we are checking for null values and we do not see any.



```
In [1346]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

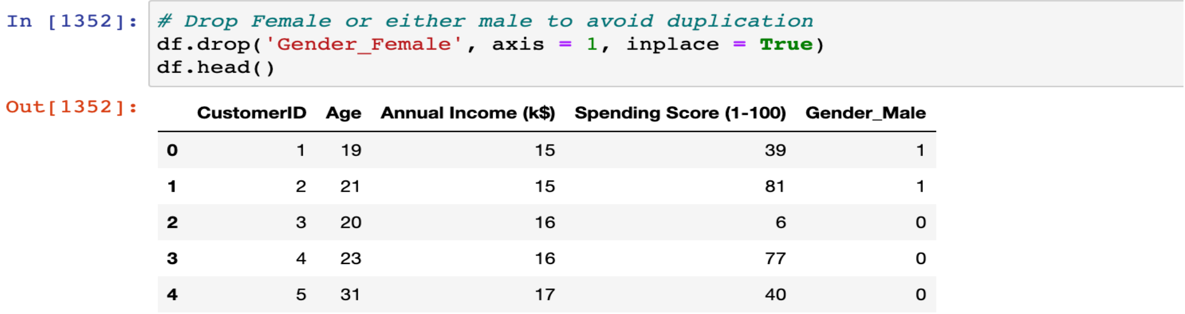
We can see that all columns are integers and Gender column is object type. So, let’s convert the object into integer.



```
In [1351]: # Perform One Hot Encoding for the Gender Column
           # converting Gender column into Integer
           df = pd.get_dummies(df, columns=['Gender'])
           df.head()
```

Out[1351]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	1	19	15	39	0	1
1	2	21	15	81	0	1
2	3	20	16	6	1	0
3	4	23	16	77	1	0
4	5	31	17	40	1	0

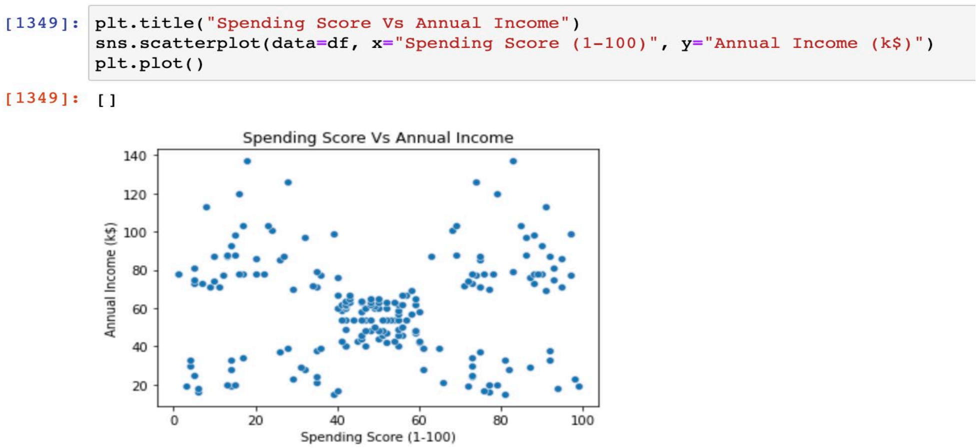


```
In [1352]: # Drop Female or either male to avoid duplication
           df.drop('Gender_Female', axis = 1, inplace = True)
           df.head()
```

Out[1352]:

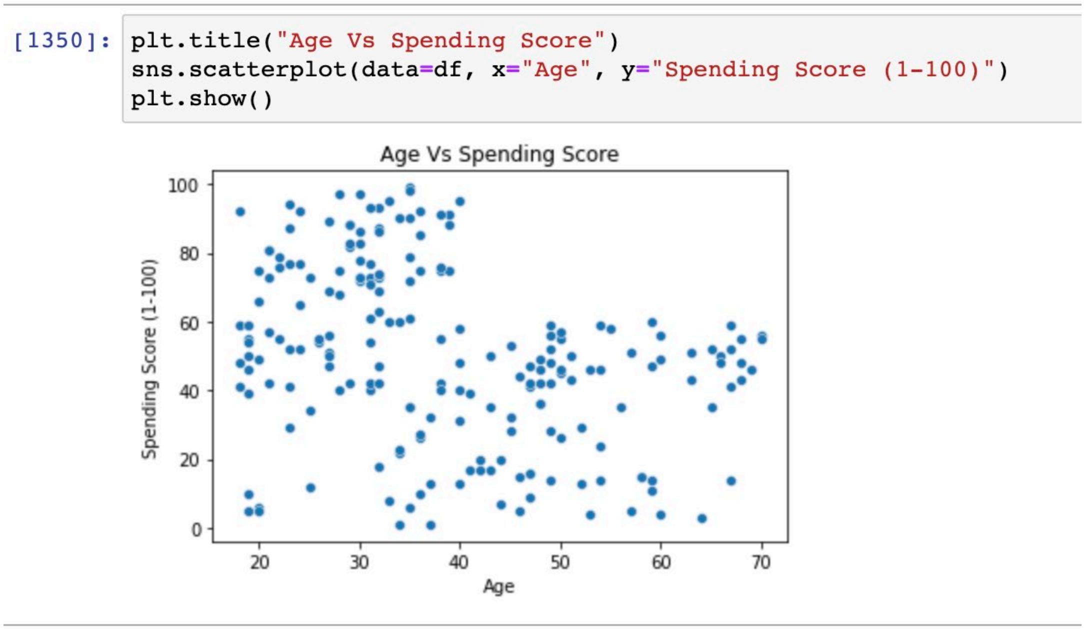
	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Male
0	1	19	15	39	1
1	2	21	15	81	1
2	3	20	16	6	0
3	4	23	16	77	0
4	5	31	17	40	0

Now, 1 – represents that the customer is male and 0 - represents that the customer is female.



```
[1349]: plt.title("Spending Score Vs Annual Income")
        sns.scatterplot(data=df, x="Spending Score (1-100)", y="Annual Income (k$)")
        plt.plot()
[1349]: []
```

By plotting the graph of Spending Score Vs Annual Income, we can conclude that Data can be divided into 5 clusters according to the income and spending score.



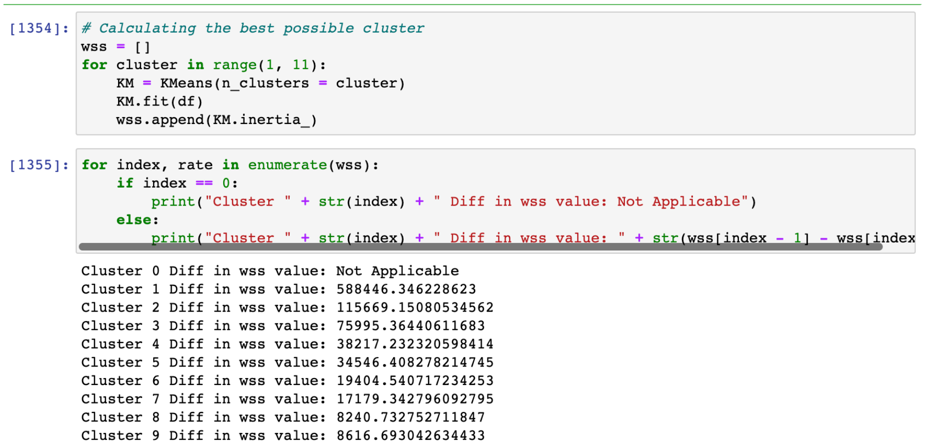
```
[1350]: plt.title("Age Vs Spending Score")
        sns.scatterplot(data=df, x="Age", y="Spending Score (1-100)")
        plt.show()
```

We can infer from the above plot that people in the 20–40 age range are more likely to spend because they are in their prime and have high disposable income.

MODELING

As the data is separated into clusters KMeans clustering is the best suitable machine learning algorithm.

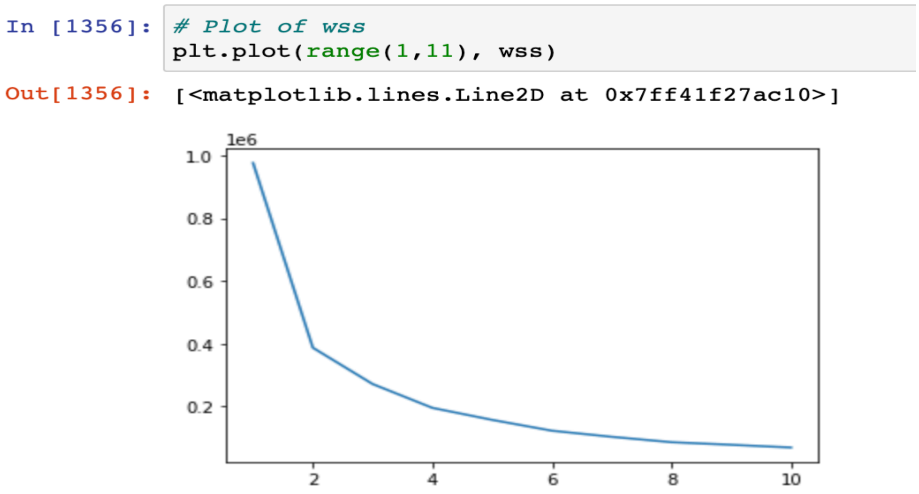
To find the optimal number of clusters lets calculate WSS Within Sum of Squares which is a technique to find K(number of clusters).



```
[1354]: # Calculating the best possible cluster
        wss = []
        for cluster in range(1, 11):
            KM = KMeans(n_clusters = cluster)
            KM.fit(df)
            wss.append(KM.inertia_)
```

```
[1355]: for index, rate in enumerate(wss):
            if index == 0:
                print("Cluster " + str(index) + " Diff in wss value: Not Applicable")
            else:
                print("Cluster " + str(index) + " Diff in wss value: " + str(wss[index - 1] - wss[index
```

```
Cluster 0 Diff in wss value: Not Applicable
Cluster 1 Diff in wss value: 588446.346228623
Cluster 2 Diff in wss value: 115669.15080534562
Cluster 3 Diff in wss value: 75995.36440611683
Cluster 4 Diff in wss value: 38217.232320598414
Cluster 5 Diff in wss value: 34546.408278214745
Cluster 6 Diff in wss value: 19404.540717234253
Cluster 7 Diff in wss value: 17179.342796092795
Cluster 8 Diff in wss value: 8240.732752711847
Cluster 9 Diff in wss value: 8616.693042634433
```

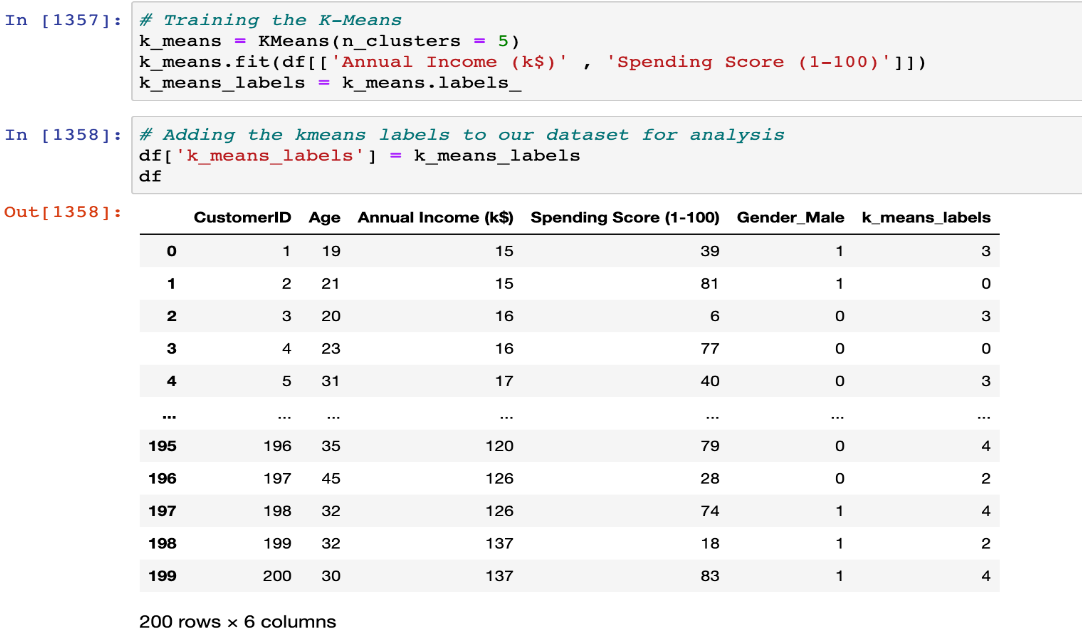


```
In [1356]: # Plot of wss
           plt.plot(range(1,11), wss)

Out[1356]: [<matplotlib.lines.Line2D at 0x7ff41f27ac10>]
```

So, by elbow plot we can determine that 5 clusters are best for this data.

Here, the machine learning model is trained with the data which is obtained from previous step.

Training the model

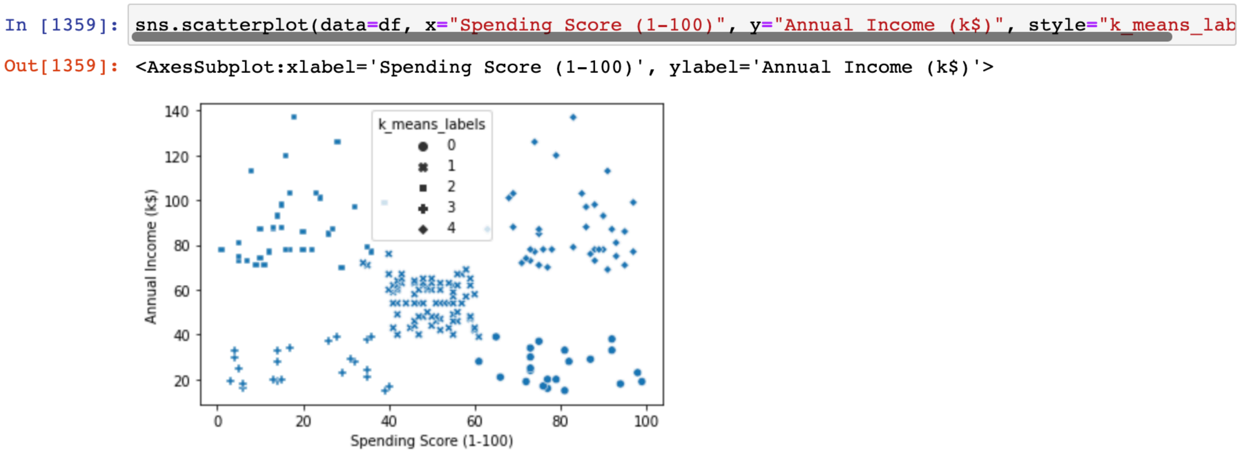


```
In [1357]: # Training the K-Means
           k_means = KMeans(n_clusters = 5)
           k_means.fit(df[['Annual Income (k$)' , 'Spending Score (1-100)']])
           k_means_labels = k_means.labels_

In [1358]: # Adding the kmeans labels to our dataset for analysis
           df['k_means_labels'] = k_means_labels
           df
Out[1358]:
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Male	k_means_labels
0	1	19	15	39	1	3
1	2	21	15	81	1	0
2	3	20	16	6	0	3
3	4	23	16	77	0	0
4	5	31	17	40	0	3
...	...	...	...	...	...	...
195	196	35	120	79	0	4
196	197	45	126	28	0	2
197	198	32	126	74	1	4
198	199	32	137	18	1	2
199	200	30	137	83	1	4

200 rows × 6 columns

Plotting the data based on clusters



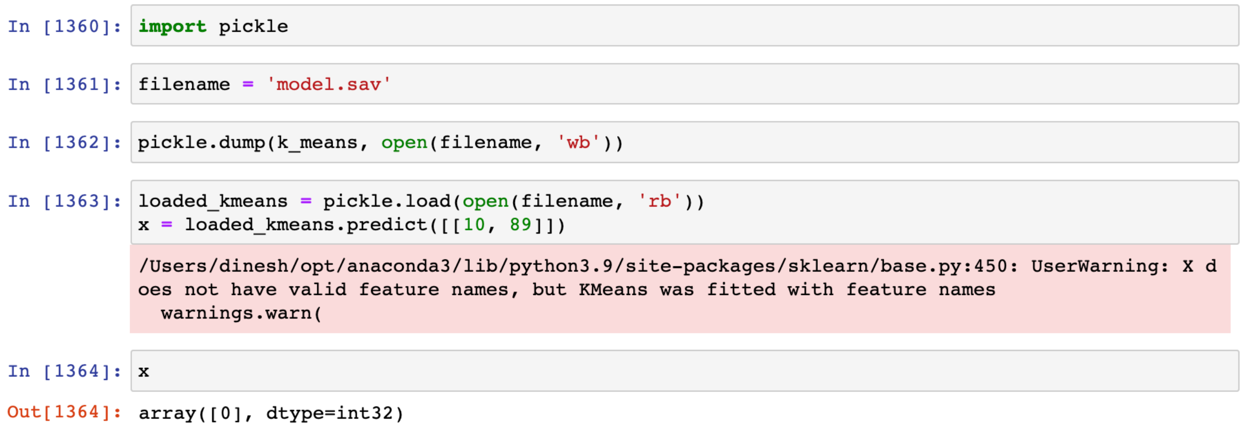
```
In [1359]: sns.scatterplot(data=df, x="Spending Score (1-100)", y="Annual Income (k$)", style="k_means_lab
Out[1359]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Annual Income (k$)'>
```

```
Category 0 –  Low Income High Spending Category
Category 1 –  Medium Income Medium Spending
Category 2 –  High Income Low Spending Category
Category 3 –  Low Income Low Spending Category
Category 4 –  High Income High Spending Category
```

EVALUATION

Once the machine learning model is trained, they are tested using random input data. We will be checking if the model positions the customer in the correct cluster(group) or not.

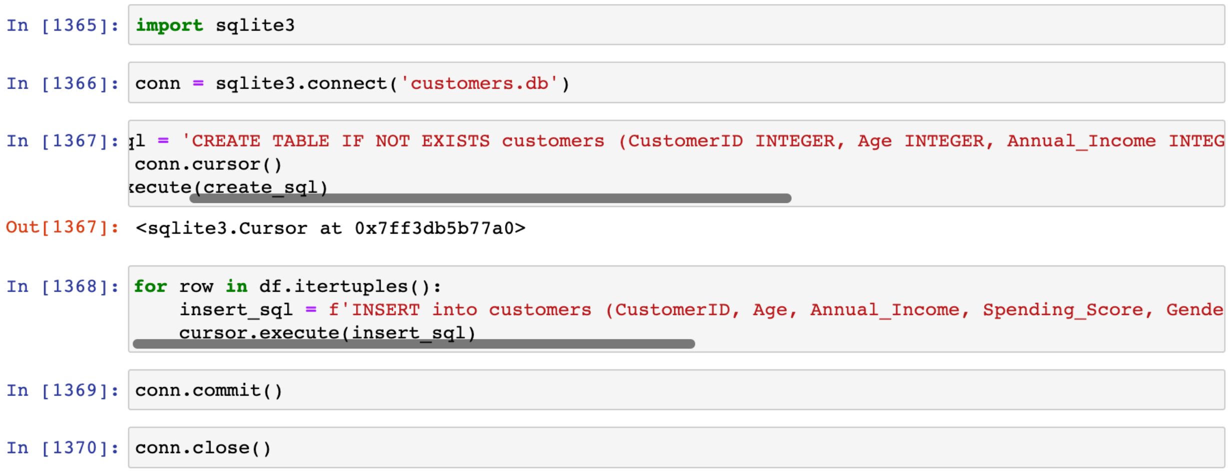
Before that let's export our trained model using pickle library.



```
In [1360]: import pickle

In [1361]: filename = 'model.sav'

In [1362]: pickle.dump(k_means, open(filename, 'wb'))

In [1363]: loaded_kmeans = pickle.load(open(filename, 'rb'))
           x = loaded_kmeans.predict([[10, 89]])

           /Users/dinesh/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X d
           oes not have valid feature names, but KMeans was fitted with feature names
             warnings.warn(

In [1364]: x

Out[1364]: array([0], dtype=int32)
```

0 specifies that this customer belongs to Category 0.

Here, we added another step to store the data of customers in database. For this we used sqlite.

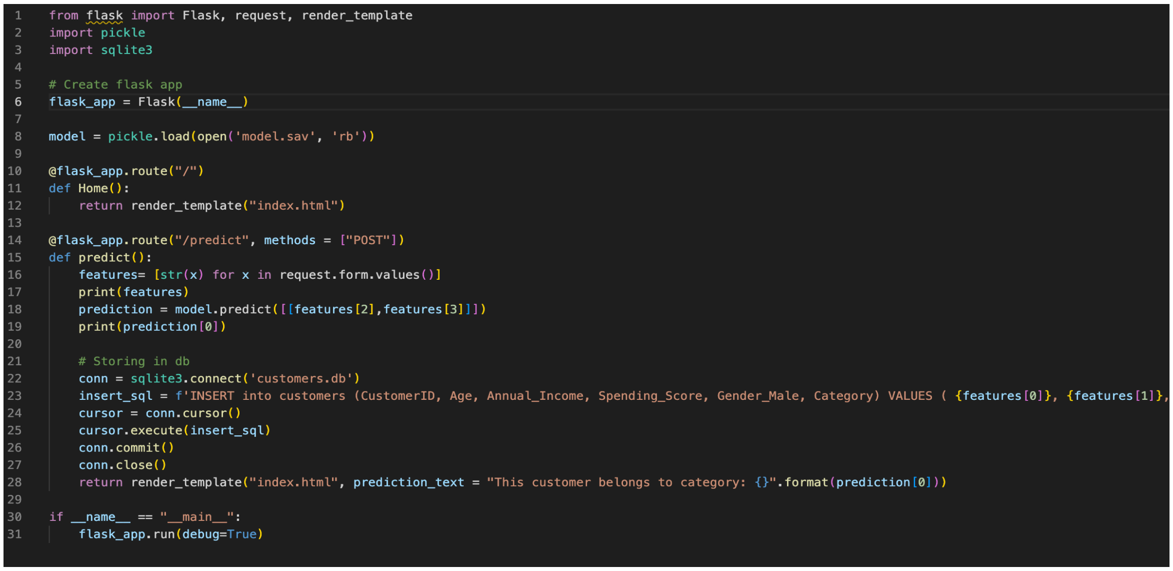


```
In [1365]: import sqlite3

In [1366]: conn = sqlite3.connect('customers.db')

In [1367]: ql = 'CREATE TABLE IF NOT EXISTS customers (CustomerID INTEGER, Age INTEGER, Annual_Income INTEG
           conn.cursor()
           xecute(create_sql)

Out[1367]: <sqlite3.Cursor at 0x7ff3db5b77a0>

In [1368]: for row in df.itertuples():
               insert_sql = f'INSERT into customers (CustomerID, Age, Annual_Income, Spending_Score, Gende
               cursor.execute(insert_sql)

In [1369]: conn.commit()

In [1370]: conn.close()
```

DEPLOYMENT

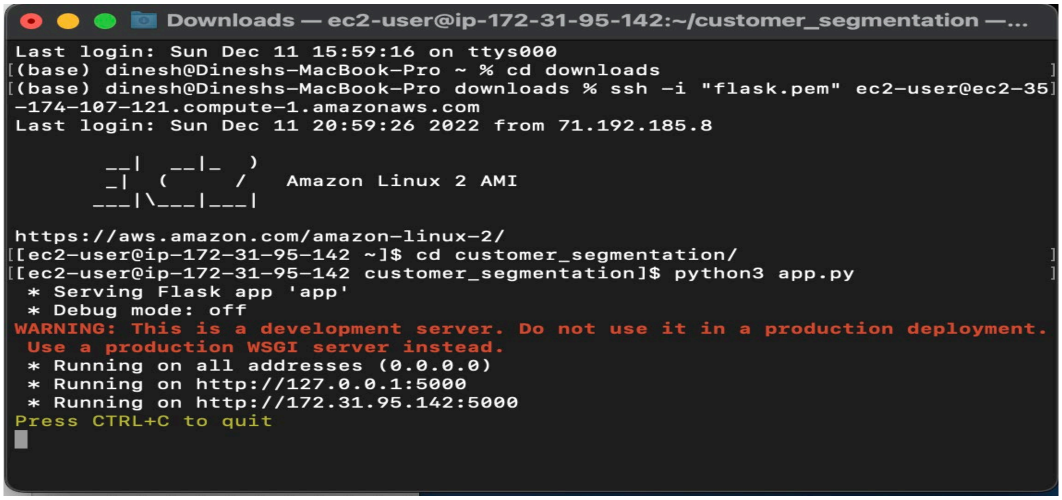
Using this trained model and database we developed an application which takes input as user data and predicts the category of the customer. Further, the input is stored in the database.

For the deployment we used Flask API



```python
from flask import Flask, request, render_template
import pickle
import sqlite3

# Create flask app
flask_app = Flask(__name__)

model = pickle.load(open('model.sav', 'rb'))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST"])
def predict():
    features= [str(x) for x in request.form.values()]
    print(features)
    prediction = model.predict([[features[2],features[3]]])
    print(prediction[0])

    # Storing in db
    conn = sqlite3.connect('customers.db')
    insert_sql = f'INSERT into customers (CustomerID, Age, Annual_Income, Spending_Score, Gender_Male, Category) VALUES ( {features[0]}, {features[1]},
    cursor = conn.cursor()
    cursor.execute(insert_sql)
    conn.commit()
    conn.close()
    return render_template("index.html", prediction_text = "This customer belongs to category: {}".format(prediction[0]))

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

The whole application is hosted in ec2 instance; hence we can access this application remotely.

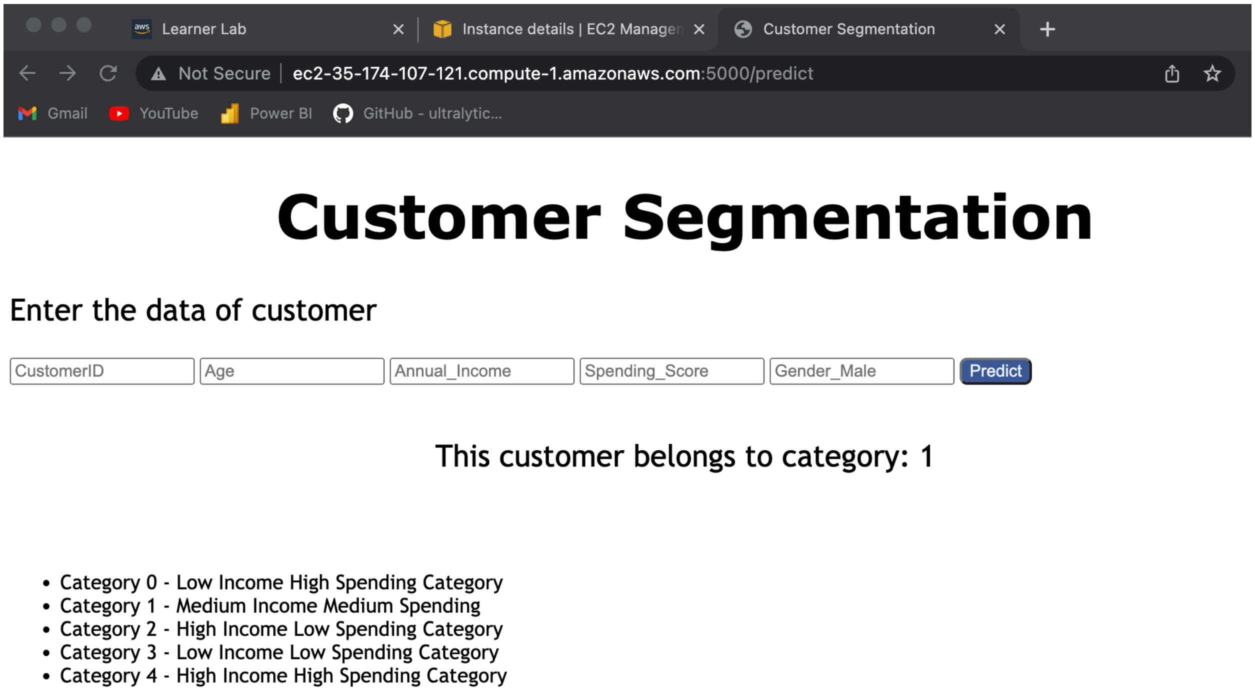


Downloads — ec2-user@ip-172-31-95-142:~/customer_segmentation —...

```
Last login: Sun Dec 11 15:59:16 on ttys000
[(base) dinesh@Dineshs-MacBook-Pro ~ % cd downloads                        ]
[(base) dinesh@Dineshs-MacBook-Pro downloads % ssh -i "flask.pem" ec2-user@ec2-35]
-174-107-121.compute-1.amazonaws.com
Last login: Sun Dec 11 20:59:26 2022 from 71.192.185.8

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[[ec2-user@ip-172-31-95-142 ~]$ cd customer_segmentation/                  ]
[[ec2-user@ip-172-31-95-142 customer_segmentation]$ python3 app.py         ]
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.31.95.142:5000
Press CTRL+C to quit
```

RESULTS

Below is a snapshot of our website where user can give the input data and internally our trained model predicts into which category the user falls in.



Customer Segmentation

Enter the data of customer

This customer belongs to category: 1

- Category 0 - Low Income High Spending Category
- Category 1 - Medium Income Medium Spending
- Category 2 - High Income Low Spending Category
- Category 3 - Low Income Low Spending Category
- Category 4 - High Income High Spending Category

REFERENCES

https://www.kaggle.com/code/fabiendaniel/customer-segmentation

https://towardsdatascience.com/customer-segmentation-with-machine-learning-a0ac8c3d4d84