Customer Segmentation Market Index

"Project Report"

Submitted by:

NAME: **M. JAWAD AHSAN**

ROLL NO: **BSAIM-F23-043**

SECTION: **AI-3A**

Submitted to:

Teacher: **Sir Rasikh Ali**

Subject: **AI(Lab)**

Table of Content

Contents

1 Introduction:

This is a machine learning project. In which I am working. "***Customer Segmentation Market Index***". This project is trained on dataset which have 195 entries and 8 columns. Further working in this project is explained in Methodology and other sections.

1.1 Background:

Normally these types of data sets are used in customer fraud detection, segmentation market index, checking customer purchase history and customer products rating. So this is a basic work on this typo working.

1.2 Objective:

We discussed in upper part, this is a basic machine learning project so in this project we working on predicting the index of the customer. We apply two models on this dataset Regression and Forest Regression and then we calculate MSE.

2 Dataset Information:

This dataset based on 195 rows and 8 columns. There are some columns I remove because these are not useable in my project. Columns name are Variable, Value, Description, Customer Count, Customer Percent, Market base count, Market base percent, and Index.

- The dataset focuses on demographic and market segmentation data, particularly customer age groups.
- There are some incomplete descriptions in the "Description" column.
- Most columns store numerical data as strings (e.g., counts and percentages), which are convert for further working.

This is the data form that we used this is not filtered data there are some columns and rows are we drop due to their unimportance and for good model training.

3 Project Workflow:

In this project we used Google Colab as an IDE. We import data frame from My-drive folder named as Customer Segmentation project. We work on google colab so there is the method of importing dataset is different from others IDE like Visual code.

The project working if we want to dividing we divide it 06 parts. The 06 parts are given below:

- Loading Dataset.
- Data Exploration.
- Preprocessing.
- Splitting.
- Training / Apply Classifiers.
- Testing & Processing Result.

These are the components are used in our project so we say that 06 parts there are some parts are further available for working.

3.1 Import Libraries:

- Pandas as pd.
- Numpy as np.
- Sklearn
- Sklearn.matrics
- Sklearn.linear model
- Sklearn.model_selection

4 Working Methodology:

We discussed the working methodology in the 3rd part. That part we discussed further in this part.

4.1 Loading dataset:

We import dataset in google colab so there are some working is required for this:

- We import library google colab and from this we import drive.
  drive.mount('/content/drive') -> By using this code we connect notebook with drive.
- The we give path of the dataset and then load this.
- The code and there output is:

```
# data_loading...
# data_view...
dataset_path = "/content/drive/My Drive/mine_file/Report.csv"
df = pd.read_csv(dataset_path)
df.head()
```

	Variable	Value	Description	Customer Count	Customer Percent	Market Base Count	Market Base Percent	Index
0	AGE CODE	A	18-24	101	1.43%	502	1.30%	109
1	AGE CODE	B	25-29	186	2.63%	1,023	2.66%	99
2	AGE CODE	C	30-34	370	5.23%	2,110	5.48%	95
3	AGE CODE	D	35-39	533	7.53%	2,877	7.48%	101
4	AGE CODE	E	40-44	614	8.68%	3,725	9.68%	90

4.2 Data Exploration:

In this part we explore data. there are methods are available by the pandas library that we used for it. There are some discussed are given below:

- We used head(), info(), describe(), shape() function normally for explore data that gives detail about our dataset. And we judge which type of data exist in dataset on which we working.
- By using these function we understand that the data have non-null values, data is in string form and the rows in data are 195 and columns are 8.

4.3 Preprocessing:

In this part the data that we explore we used and remove all columns, rows and cells that are not required. Why we perform preprocessing is a question that occurs at this part, the answer is to increase accuracy of dataset, and make some operations like testing and etc.

In this project we drop Variable columns that are unnecessary and after this we drop market base count and customer count.

4.3.1 Example:

```
df = df.drop("Variable", axis=1)
```

```
df["Index"] = df["Index"].str.replace(',', '').str.strip()

# Step 2: Convert to numeric, setting invalid values to NaN
df["Index"] = pd.to_numeric(df["Index"], errors='coerce')

# Step 3: Optionally handle NaN values (e.g., replace with 0 or drop them)
df["Index"] = df["Index"].fillna(0)
```

This a example of preprocessing we drop variable columns.

4.4 Splitting:

After preprocessing normally data is divided in two parts one part is for training and the second part is for testing. The training part data is normally 70 or 80 percent and for testing 20 and 30 percent of data is used. In this part we divide data in 80, 20. We used 80% for training and 20% for testing.

Training data set view is given below:

```
x_train.head()
```

	Value	Description	Customer Percent	Market Base Percent
5	F	45-49	8.92	9.66
135	O	FIFTEENTH RANK	2.59	3.36
122	B	SECOND RANK	14.26	13.67
167	5	41 - 50	11.69	13.75
85	I	$500,000 TO $1 MILLION 524	7.41	6.99

4.5 Training:

The dataset includes both categorical and numerical features. Categorical columns ("Value" and "Description") undergo One-Hot Encoding (OHE) via a ColumnTransformer to ensure they are suitable for modeling. The remaining features are passed through unchanged. This transformation is applied to both the training and test sets, ensuring consistency. The transformed data is then converted into DataFrames for better interpretability.

4.6 Machine Learning Model

4.6.1 Linear Regression

A baseline model that assumes a linear relationship between inputs and the target. It is trained on the preprocessed training data and evaluated using Mean Squared Error (MSE) and R-squared (R^2) metrics.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train_dataset, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test_dataset)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 4.024445617404648e+28
R-squared: -3.292425626804665e+23
```

4.6.2 Random Forest Regression

An ensemble model that builds multiple decision trees, capturing non-linear relationships and reducing overfitting. It is trained on the transformed data and evaluated similarly with MSE and R^2. Random Forest often outperforms simpler models like Linear Regression due to its ability to model complex patterns.

```
from sklearn.ensemble import RandomForestRegressor
# Initialize and train the Random Forest model
rf_model = RandomForestRegressor(random_state=42)  #
rf_model.fit(X_train_dataset, y_train)
# Make predictions on the test set
rf_y_pred = rf_model.predict(X_test_dataset)
# Evaluate the model
rf_mse = mean_squared_error(y_test, rf_y_pred)
rf_r2 = r2_score(y_test, rf_y_pred)
print(f"Random Forest Mean Squared Error: {rf_mse}")
print(f"Random Forest R-squared: {rf_r2}")
```

```
Random Forest Mean Squared Error: 389598.6852743589
Random Forest R-squared: -2.1873326602284404
```

5 Conclusion:

This project involved preprocessing a dataset with One-Hot Encoding and a ColumnTransformer to prepare it for machine learning models. Linear Regression was used as a baseline, offering simplicity but limited performance in complex scenarios. The Random Forest Regressor, leveraging its ability to handle non-linear patterns, outperformed Linear Regression in both accuracy (lower MSE) and explanatory power (higher R^2). This highlights the significance of model selection based on the complexity of the dataset.