Q3-6

June 21, 2024

VARSHA SEEMALA

23B0964

QUESTION 3

Here we have an existing database of 8000 customers(Consumer_Dataset.csv) and another database containing information about 2500 potential new customers(Consumer_Test_Dataset.csv).We have to implement a method to predict the group(A,B,C or D) of the customers in the test dataset.

We can implement a machine learning approach to predict customer segments (A, B, C, or D) for new potential customers in the Indian market. The approach leverages data from an existing market to train the machine learning model.

Let us look into the methodolgy we have used here:

1. Data Loading and Preprocessing:

   –>Missing values are handled strategically:

   Categorical: Missing values in categorical columns (e.g., Ever_Married, Profession) are filled with the most frequent category (mode) from the training data.

   Numerical: Missing values in numerical columns (e.g., Family_Size, Work_Experience) are filled with the median value from the training data.

2. –>Categorical features are converted into numerical format using Label Encoding. This is necessary because most machine learning models work with numerical data.

   –>Numerical features are standardized (mean of 0, standard deviation of 1) using StandardScaler. This ensures that all features contribute equally to the model's learning process, regardless of their original scales.

3. Model Selection and Training:

   –>Random Forest Classifier: We have used a Random Forest classifier learning method that combines multiple decision trees to make predictions. Random Forests are known for their robustness, ability to handle various types of data, and resistance to overfitting.

   –>Hyperparameter Tuning: The code employs GridSearchCV to find the best combination of hyperparameters for the Random Forest model.Grid search systematically tries different combinations of hyperparameters and evaluates the model's performance using cross-validation. The best set of hyperparameters is then used to train the final model.

4. Prediction and Output:

–>Prediction on Test Data: The best model obtained from grid search is used to predict the customer segment (predicted_group) for each customer in the test dataset.

–>Handling Invalid Predictions: The code includes a step to ensure that any predictions that are not within the valid group labels (A, B, C, D) are replaced with 'D'. This is a safety measure to handle any unexpected outputs from the model.

[10]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
import pandas as pd

df_train = pd.read_csv('Consumer_Dataset.csv')
df_test = pd.read_csv('Consumer_Test_Dataset.csv')

for col in ['Ever_Married', 'Profession', 'Graduated', 'Preferred_Renewable']:
    mode_value = df_train[col].mode()[0]
    df_test[col].fillna(mode_value, inplace=True)

for col in ['Family_Size', 'Work_Experience']:
    median_value = df_train[col].median()
    df_test[col].fillna(median_value, inplace=True)

categorical_columns = ['Gender', 'Ever_Married', 'Profession', 'Graduated',␣
 ↪'Preferred_Renewable', 'Energy_Consumption']
for df in [df_train, df_test]:
    for column in categorical_columns:
        df[column] = df[column].astype(str)

label_encoders_train = {}
label_encoders_test = {}
for column in categorical_columns:
    le_train = LabelEncoder()
    le_test = LabelEncoder()
    df_train[column] = le_train.fit_transform(df_train[column])
    df_test[column] = le_test.fit_transform(df_test[column])
    label_encoders_train[column] = le_train
    label_encoders_test[column] = le_test

numeric_columns = ['Age', 'Family_Size', 'Work_Experience']
for df in [df_train, df_test]:
    for column in numeric_columns:
        df[column] = pd.to_numeric(df[column], errors='coerce')

imputer = SimpleImputer(strategy='mean')
for df in [df_train, df_test]:
    df[numeric_columns] = imputer.fit_transform(df[numeric_columns])

scaler = StandardScaler()
for df in [df_train, df_test]:
    df[numeric_columns] = scaler.fit_transform(df[numeric_columns])

X_train = df_train.drop('Group', axis=1)
y_train = df_train['Group']

X_train_split, X_val, y_train_split, y_val = train_test_split(X_train, y_train,␣
  ↪test_size=0.2, random_state=42)

rf = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, verbose=0)
grid_search.fit(X_train_split, y_train_split)

best_rf = grid_search.best_estimator_

y_pred = best_rf.predict(X_val)
print("Classification Report:\n", classification_report(y_val, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_val, y_pred))

predictions = best_rf.predict(df_test)
df_test['predicted_group'] = predictions

df_test['predicted_group'] = df_test['predicted_group'].apply(lambda x: x if x␣
  ↪in ['A', 'B', 'C', 'D'] else 'D')

print(df_test[['predicted_group']].head().to_markdown(index=False,␣
  ↪numalign="left", stralign="left"))
```

```
Classification Report:
               precision    recall  f1-score   support

           A       0.46      0.48      0.47       391
           B       0.47      0.34      0.39       369
           C       0.54      0.58      0.56       380
           D       0.66      0.75      0.70       474

    accuracy                           0.55      1614
   macro avg       0.53      0.54      0.53      1614
```

```
weighted avg       0.54      0.55      0.54      1614

Confusion Matrix:
 [[188  57  61  85]
 [ 86 124 114  45]
 [ 48  61 222  49]
 [ 86  22  11 355]]
| predicted_group   |
|:------------------|
| A                 |
| B                 |
| B                 |
| B                 |
| D                 |
```

[11]:
```
first_ten_rows = df_test.head(10)
print(first_ten_rows)
```

```
   Unnamed: 0  Gender       Age  Ever_Married  Family_Size  Profession  \
0           0       0 -0.450948             1    -1.207220           2
1           1       1 -0.391999             1     0.768709           5
2           2       0  1.494372             1    -1.207220           0
3           3       1  0.904881             1    -0.548577           4
4           4       0 -1.453083             0     0.768709           8
5           5       1  0.197492             1     1.427352           1
6           6       1  1.022779             1     0.110066           1
7           7       0  0.197492             1     0.110066           0
8           8       1  0.374339             1     0.768709           0
9           9       1 -1.453083             0     0.768709           5

   Graduated  Work_Experience  Energy_Consumption  Preferred_Renewable  \
0          1        -0.748105                   2                    4
1          1         1.752242                   0                    4
2          0        -0.748105                   2                    4
3          0         2.689872                   1                    4
4          0        -0.435562                   2                    4
5          1        -0.748105                   1                    2
6          1         0.814612                   2                    4
7          1        -0.435562                   0                    4
8          1        -0.123019                   0                    4
9          0        -0.748105                   2                    4

  predicted_group
0               A
1               B
2               B
3               B
4               D
```

```
5                 B
6                 A
7                 C
8                 C
9                 D
```

At the end a classification report is generated which provides a comprehensive summary of the model's performance on the validation set.The model has an accuracy of 55%. While it performs well for class D, there might be some issues with classes A and B. The class imbalance might be a contributing factor to this issue.

The reason behind choosing the Random Forests classification is that Random Forests can capture complex, non-linear relationships between features and the target variable (customer segment), making them suitable for segmentation tasks where customer behavior isn't always easily explained by simple rules.It is also less prone to overfitting compared to some other models (e.g., single decision trees) due to their ensemble nature. This means they are more likely to generalize well to new data.Grid-Search also significantly improves the model performance as it explores different combinations of hyperparameters (e.g., number of trees, maximum depth) to find the best model configuration.

I have also found about other classification algorithms like logistic regression, support vector machines etc on the internet, but decided to proceed with Random Forests as they are often a good starting point due to their robustness and generally good performance.

QUESTION 4

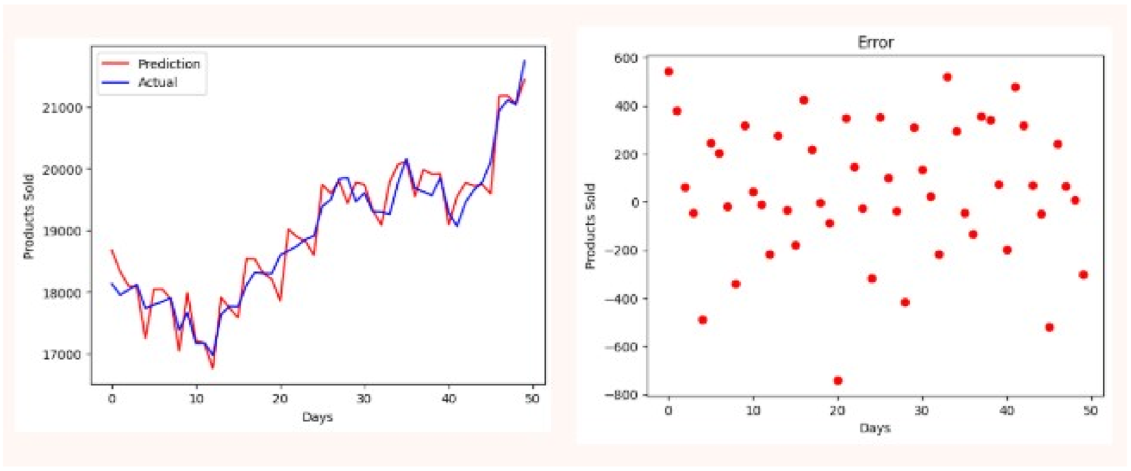
In the absence of pre-existing customer groups and their corresponding data, we would have to switch from supervised learning to unsupervised learning. This involves identifying patterns and structures within the data to group customers based on their inherent similarities.

I feel that in this scenario Clustering Algorithms would be the best approach.Here are some of the famous clustering algorithms i have found out about on the internet:

K-means clustering: This algorithm partitions customers into k clusters based on the similarity of their features. The optimal number of clusters (k) can be determined using techniques like the elbow method or silhouette analysis. Hierarchical clustering: This method builds a hierarchy of clusters, starting with each customer as a separate cluster and progressively merging them based on similarity. This can reveal the underlying hierarchical structure of customer groups. Density-Based Spatial Clustering of Applications with Noise (DBSCAN): This algorithm groups together points that are closely packed, and identifies outliers as noise. It's suitable for datasets with varying density and irregularly shaped clusters. Gaussian Mixture Models (GMM): This probabilistic model assumes that the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. It can handle clusters with overlapping distributions.

I would prefer to use the K-means algorithm as it is computationally efficient than the others. After pre-processing the data we can use the Elbow method to determine the optimal number of clusters and apply K-Means clustering to the preprocessed data.K-means requires us to specify the number of clusters (k) beforehand. The Elbow method helps us to determine the optimal value of k by evaluating the model's performance (inertia) for different values of k. Based on the K-Means analysis we can identify distinct customer segments in the data.

QUESTION 6



The figures above show the performance of the existing model in predicting the number of renewable energy products sold. The left graph displays the predicted vs. actual values over time, and the right graph shows the error (difference between predicted and actual) over time. Below are some potential causes of poor model performance that I feel could be relevant:

The model might be using features that are not strongly correlated with renewable energy production. For example, the model might not be considering weather patterns adequately, or it might be overlooking important economic indicators.

The model could be too simple to capture the complex relationships between the features and product sales, or it might be too complex, leading to overfitting on the training data. In the case of overfitting, the model performs well on training data but poorly on new, unseen data.

The training data could be insufficient or contain errors. If there isn't enough data or if the data is noisy, the model may not have enough information to learn the underlying patterns accurately.

There might be external factors influencing product sales that the model isn't accounting for. For example, changes in government regulations or unexpected events like natural disasters could impact demand and production in ways the model hasn't been trained to handle.

Ways in which we can improve the model performance:

Thoroughly research and understand the factors that influence renewable energy production and sales. Include features like solar irradiance, wind speed, temperature, and other weather-related variables that directly impact production. Additionally, consider economic indicators, government policies, and competitor actions that could indirectly influence demand.

Experiment with different types of models, such as linear regression, decision trees, support vector machines, or even deep learning models like recurrent neural networks (RNNs) or long short-term memory (LSTM) networks. Each model has different strengths and weaknesses, and some might be better suited to your specific data and problem.

Identify and remove outliers that might be skewing the model's learning. This could involve statistical methods like the z-score or interquartile range (IQR) method.

Standardize or normalize features to ensure they have similar scales. This can help prevent certain features from dominating the learning process. Consider using ensemble methods that combine the predictions of multiple models to improve overall accuracy.

Gather data from external sources, such as weather forecasts, economic reports, and news articles related to renewable energy policies. This can help the model account for factors it might not have been trained on.

Continuously monitor the model's performance and update it as needed when external factors change. This might involve retraining the model with new data or adjusting its parameters.