Created 12th August 2024

Bertelsmann/Arvato Project

Project Definition

Project Overview

The primary objective of this Capstone Project is to develop a comprehensive customer segmentation report for Arvato Financial Services by analyzing demographic data of customers from a mail-order sales company in Germany. This analysis will compare the customer data against the broader demographics of the general population to identify distinctive segments that best represent the company's core customer base. Then, this project will build a model to make predictions on whether a person will be a customer or not based on the demographic data.

Scope of Work

- Data Analysis: Leveraging advanced data analytics, this project will delve into extensive demographic datasets to understand the characteristics that differentiate the company's customers from the general population. This involves using unsupervised learning techniques such as clustering to segment the customer base.
- Segmentation Application: The insights gained from the initial segmentation will then be applied to a separate dataset comprising potential targets of a marketing campaign. The goal here is to identify the customer segments matching with the general population
- Predictive Modeling: Utilizing the segmentation insights, participants will develop predictive models to estimate the likelihood of conversion for the individuals targeted in the marketing campaign. This step involves selecting appropriate modeling techniques that best fit the nature of the data and the specific objectives of the project.

Problem Statement

How to precisely target individuals who are likely to convert into customers, thereby increasing the efficiency and effectiveness of marketing campaigns.

The objective is to harness advanced data analytics to develop a comprehensive customer segmentation report. This report will provide Arvato Financial Services with actionable insights by identifying unique demographic segments that accurately represent the company's core customer base.

Additionally, the project aims to leverage these insights to build a predictive model capable of forecasting an individual's likelihood of becoming a customer based on their demographic dat

Data inputs

There are four data files associated with this project:

- **`Udacity_AZDIAS_052018.csv`:** Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).

- **`Udacity_CUSTOMERS_052018.csv`:** Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).

- **`Udacity_MAILOUT_052018_TRAIN.csv`:** Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).

- **`Udacity_MAILOUT_052018_TEST.csv`:** Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

Each row of the demographics files represents a single person, but also includes information outside of individuals, including information about their household, building, and neighborhood. Use the information from the first two files to figure out how customers ("CUSTOMERS") are similar to or differ from the general population at large ("AZDIAS"), then use your analysis to make predictions on the other two files ("MAILOUT"), predicting which recipients are most likely to become a customer for the mail-order company.

The "CUSTOMERS" file contains three extra columns ('CUSTOMER_GROUP', 'ONLINE_PURCHASE', and 'PRODUCT_GROUP'), which provide broad information about the customers depicted in the file. The original "MAILOUT" file included one additional column, "RESPONSE", which indicated whether or not each recipient became a customer of the company. For the "TRAIN" subset, this column has been retained, but in the "TEST" subset it has been removed; it is against that withheld column that your final predictions will be assessed in the Kaggle competition.

Otherwise, all of the remaining columns are the same between the three data files. For more information about the columns depicted in the files, you can refer to two Excel spreadsheets provided in the workspace. [One of them](./DIAS Information Levels - Attributes 2017.xlsx) is a top-level list of attributes and descriptions, organized by informational category. [The other](./DIAS Attributes - Values 2017.xlsx) is a detailed mapping of data values for each feature in alphabetical order.

Metrics

This project is divided into two parts

Customer segmentation using unsupervised algorithms

In this phase of the project, we employ Principal Component Analysis (PCA) as a dimensionality reduction technique to streamline the dataset into a manageable number of features. The selection of the optimal number of principal components is guided by the explained variance ratio, which quantifies the amount of variance each principal component captures from the data. We aim to retain the minimum number of dimensions that still capture a substantial amount of the dataset's variation.

Following the dimension reduction, we propose using K-Means Clustering, an unsupervised learning algorithm, to segment the customer base into distinct clusters. The optimal number of clusters is determined by analyzing the elbow plot, which plots the sum of squared distances from each point to its assigned center (squared errors). This plot helps identify the point where the marginal gain in explained variance drops off, indicating the elbow point and the most appropriate number of clusters.

Customer Acquisition using supervised learning algorithms

In the second phase of the project, the objective is to predict whether the mail-order company should engage with a prospective customer. For this task, the provided training data will be divided into two subsets: a training set and an evaluation set. The model will be trained using the training set and subsequently assessed using the evaluation set. Given the nature of this task, appropriate classification evaluation metrics will be employed to gauge the model's performance.

Given these requirements, the Area Under the Receiver Operating Characteristic (AUROC) has been selected as the primary evaluation metric. The AUROC metric provides a comprehensive measure of model performance across various threshold settings by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). A model that performs well will approach an AUROC score of 1, indicating superior discriminatory ability. Therefore, a higher AUROC value signifies better model performance, which is essential for ensuring both accurate and balanced predictions in the context of this imbalanced classification challenge.

Methodology

Data Preprocessing

Addressing column-wise missing data

Column-Wise Analysis of Missing Data: We conducted a detailed examination of the missing values percentage across each column. It was observed that columns with missing values in the customer dataset frequently corresponded to columns with missing data in the general population dataset. After a thorough review of the missing value percentage distribution, a threshold of 20% was established. Columns exceeding this threshold of missing data were considered too sparse for reliable analysis and were consequently removed from both the customer and general population datasets. In total, 16 columns were eliminated during this phase of data preprocessing.

The list of columns to drop for both datasets (general population and customer dataset):
'ALTER_KIND4', 'ALTER_KIND3','ALTER_KIND2', 'ALTER_KIND1', 'EXTSEL992', 'KK_KUNDENTYP', 'ALTERSKATEGORIE_FEIN', 'D19_VERSAND_ONLINE_QUOTE_12', 'D19_LOTTO', 'D19_BANKEN_ONLINE_QUOTE_12', 'D19_LETZTER_KAUF_BRANCHE', 'D19_SOZIALES', 'D19_GESAMT_ONLINE_QUOTE_12', 'D19_KONSUMTYP', 'D19_VERSI_ONLINE_QUOTE_12', 'D19_TELKO_ONLINE_QUOTE_12'

Addressing row-wise missing data

Row-Wise Analysis of Missing Data: During this phase, we assessed the number of missing values in each row of our datasets. Any rows containing more than 50 missing features were deemed unreliable for analysis and subsequently removed. This criterion led to the elimination of 139,890 observations from the general population dataset, which initially comprised 891,211 observations. Similarly, from the customer dataset, which originally included 191,652 observations, a total of 56,508 observations were discarded due to excessive missing values.

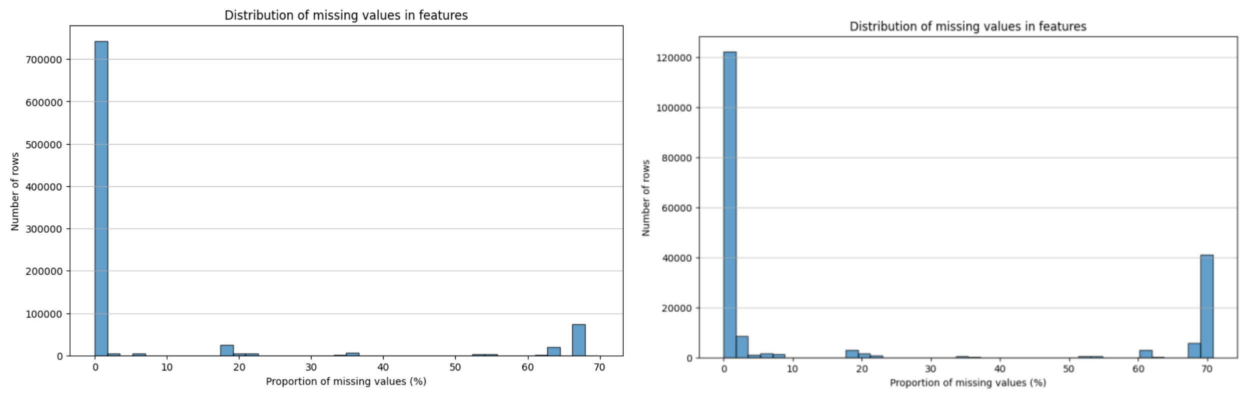


Figure 1: Distributions of missing values in each rows (left: azdias dataset, right: customer dataset)

Re-encode categorical and mixed features & drop unnecessary feature

Several features in the dataset have undergone re-encoding, transformation, and dropping to better suit our analysis:

EINGEFUGT_AM:

- Original Format: Initially recorded as a date marking when the person joined or the entry was made.
- Drop: Drop this column because we don't use this column in our analysis

CAMEO_DEU_2015:

- Original Format: categorical values and contained 46 values
- Drop: Drop this column

ANREDE_KZ:

- Original Encoding: Gender was encoded with values '1' and '2', representing male and female, respectively.
- Re-Encoding: Adjusted to a binary format where '0' represents male and '1' represents female for standardization and ease of interpretation.

CAMEO_INTL_2015, CAMEO_DEUG_2015 :

- Original Format: Contained X or XX in the values
- Transformation: Replaced X and XX with NaN

OST_WEST_KZ

- Original Format: Contained two categorical values W and O
- Re-Encoding: Map these values into Boolean value W represents 0 and O represents 1

Imputing missing values

After the initial removal of features and rows that exceeded the set thresholds for missing values, the dataset still contained some residual missing entries. To address these, we opted for a methodical approach to imputation to ensure consistency and retain as much data integrity as possible:

- **Imputation Strategy**: The remaining missing values in the dataset have been imputed using the mode (most frequently occurring observation) for each feature. This method was chosen because:
    - **Relevance to Population Data**: The dataset represents a general population, and imputing with the most common values provides a statistically robust approximation that aligns with the observed trends and distributions within the data.
    - **Preservation of Distribution**: Using the mode helps maintain the original distribution of the data, which is crucial for any subsequent analysis that relies on population characteristics.

Remove correlated columns

In this project, we implemented a correlation analysis to identify and remove highly correlated features that could potentially skew the model's interpretation of importance or cause multi-collinearity issues. Here's how we approached this:

- **Correlation Threshold**: We defined "too highly correlated" features as those having a correlation coefficient greater than 0.8 with any other feature in the dataset. This threshold helps ensure that the remaining features provide unique information without excessive redundancy.
- **Outcome of Analysis**: As a result of this stringent criterion, 62 features exhibiting high correlation were identified and subsequently removed from the dataset.

Feature scaling

A standard scaler is used to bring all the features to the same range. This is done in order to eliminate feature dominance when applying dimensionality reduction.

Implementation

In this section, we will add the implementing details of machine learning algorithms.

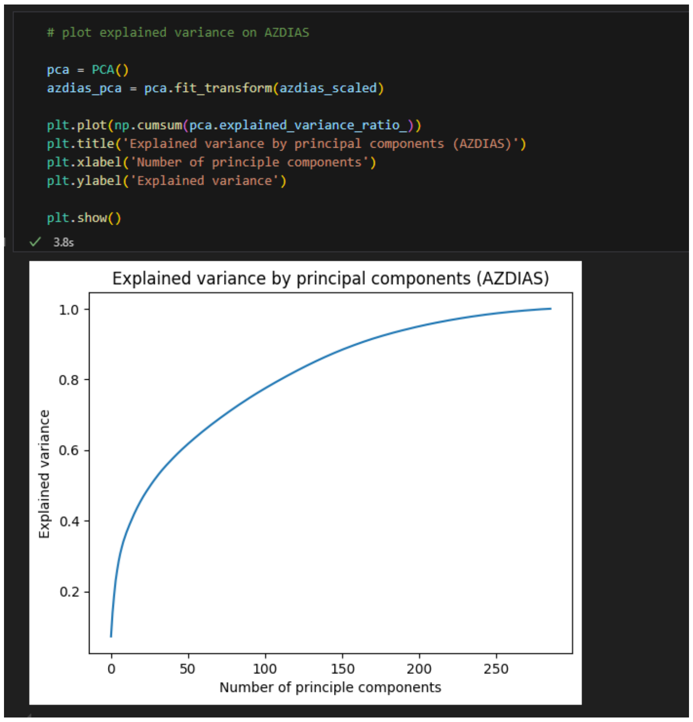
Dimension Reduction with PCA

Principal Component Analysis (PCA) is used to reduce the dimensionality of our dataset while retaining the variance present in the original data. Reducing dimensionality helps in minimizing overfitting, improves computation time, and makes the dataset easier to visualize.

High-dimensional datasets can be challenging to work with due to the "curse of dimensionality," where the complexity of the data increases exponentially with the number of features. Reducing the number of features helps to:

- Improve computational efficiency.
- Reduce noise and potential overfitting.
- Make the data easier to visualize and interpret.

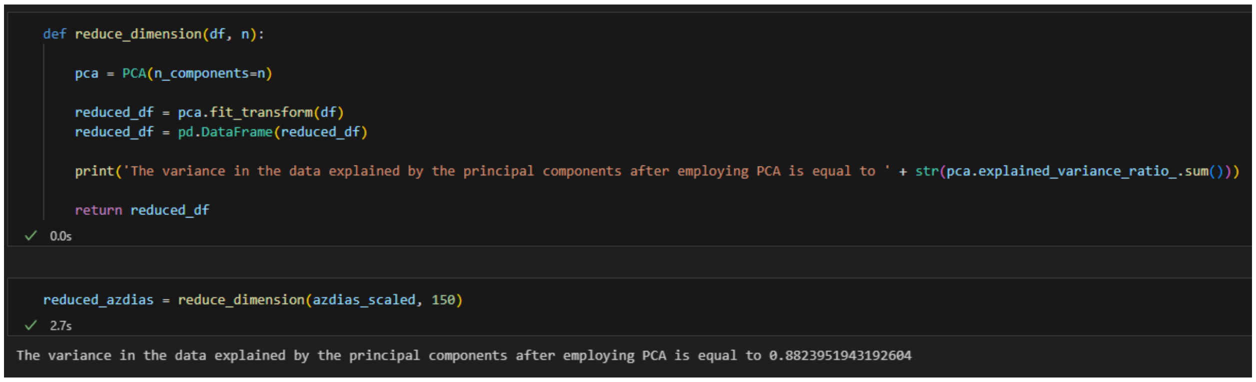
To decide the number of principal components we want to have (and thus, the final number of features we want to keep in our azdias and customers DataFrames), we used scree plot visualizations. A scree plot displays the variance explained by each principal component. The goal is to only keep the number of principal components that will explain above 80% of the variance in the data (a percentage deemed high enough that we don't lose so much information in the data). After reviewing the scree plot, we decide that 150 is a good choice for the number of principal components we want to keep.



```
# plot explained variance on AZDIAS

pca = PCA()
azdias_pca = pca.fit_transform(azdias_scaled)

plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.title('Explained variance by principal components (AZDIAS)')
plt.xlabel('Number of principle components')
plt.ylabel('Explained variance')

plt.show()
```
✓ 3.8s

Explained variance by principal components (AZDIAS)
Explained variance
1.0
0.8
0.6
0.4
0.2
0
50
100
150
200
250
Number of principle components

After examining the scree plot, we decided to keep 150 principal components, which collectively explained approximately 88% of the variance. This balance allowed us to maintain most of the data's structure while significantly reducing its complexity.



```
def reduce_dimension(df, n):

    pca = PCA(n_components=n)

    reduced_df = pca.fit_transform(df)
    reduced_df = pd.DataFrame(reduced_df)

    print('The variance in the data explained by the principal components after employing PCA is equal to ' + str(pca.explained_variance_ratio_.sum()))

    return reduced_df
```
✓ 0.0s

```
reduced_azdias = reduce_dimension(azdias_scaled, 150)
```
✓ 2.7s

```
The variance in the data explained by the principal components after employing PCA is equal to 0.8823951943192604
```

We applied this dimensionality reduction process to both the azdias and customers datasets, ensuring that the transformed data maintained its integrity. A function was created to automate the dimensionality reduction process, facilitating its application across different datasets.

Cluster with Kmeans

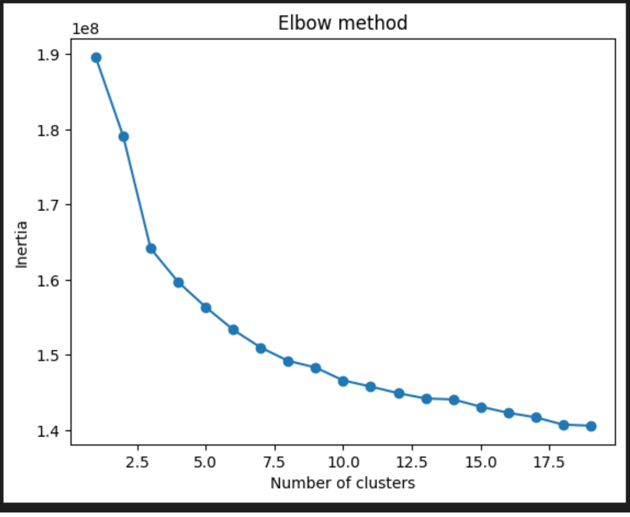
Clustering helps us identify distinct groups within our data, enabling better customer segmentation and targeted analysis. KMeans is a popular clustering algorithm due to its simplicity and efficiency, making it suitable for our large dataset.

Selecting the right number of clusters (K) is crucial for meaningful segmentation. We used the Elbow Method, which involves plotting the sum of squared errors (SSE) against the number of clusters. The "elbow" point in the graph indicates a balance between the number of clusters and the compactness of the clustering.

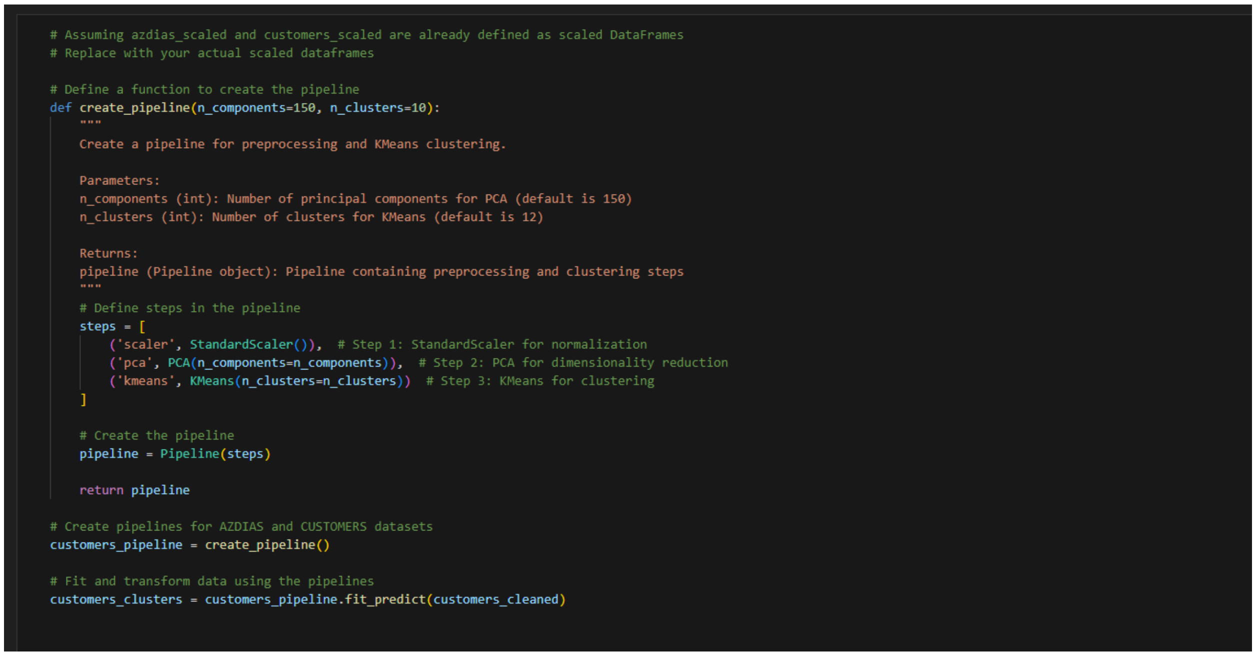
After applying the Elbow Method, we chose K=12 as the optimal number of clusters. This number provided a good balance between the granularity of the segmentation and the compactness of each cluster. We then created a pipeline that included PCA (with 150 components) and KMeans clustering, and applied it to the customers dataset

```
inertias = []
for i in range(1, 20):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(reduced_azdias)
    inertias.append(kmeans.inertia_)

plt.plot(range(1,20), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

✓ 5m 24.9s

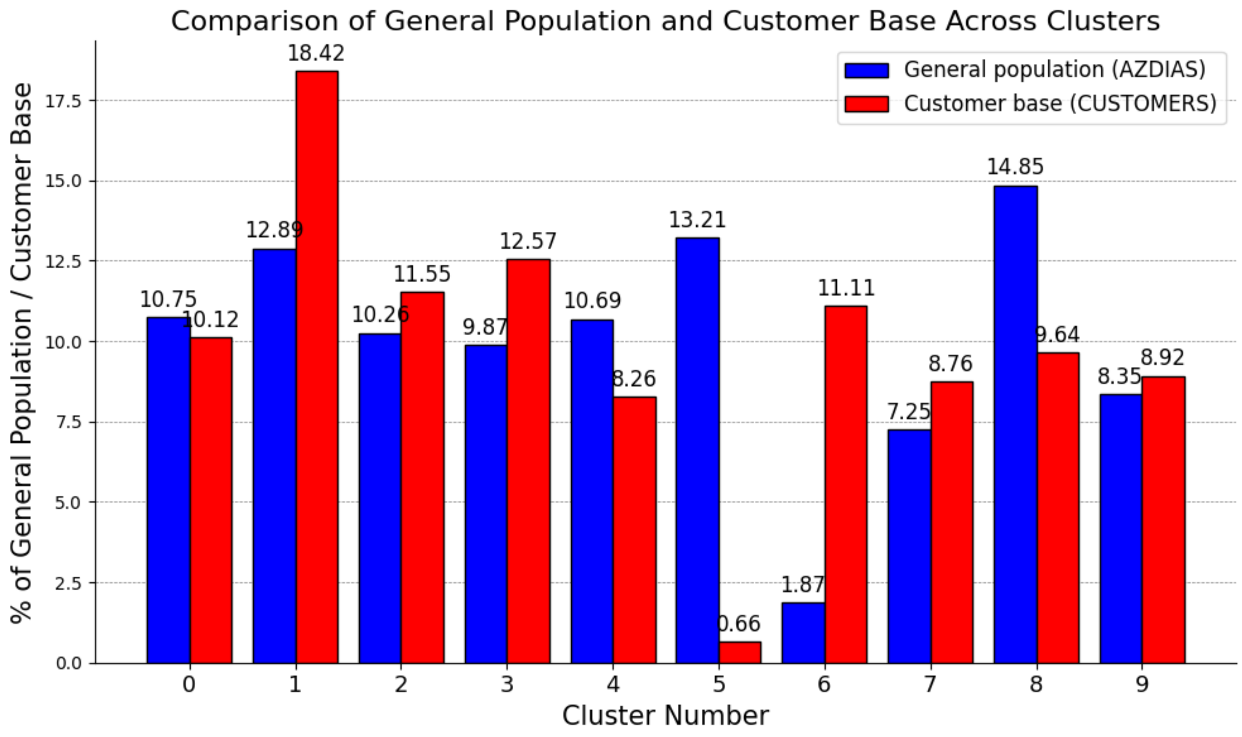


I created a pipeline to apply the same steps for customers dataset with the PCA is 150 and cluster is 12



```python
# Assuming azdias_scaled and customers_scaled are already defined as scaled DataFrames
# Replace with your actual scaled dataframes

# Define a function to create the pipeline
def create_pipeline(n_components=150, n_clusters=10):
    """
    Create a pipeline for preprocessing and KMeans clustering.

    Parameters:
    n_components (int): Number of principal components for PCA (default is 150)
    n_clusters (int): Number of clusters for KMeans (default is 12)

    Returns:
    pipeline (Pipeline object): Pipeline containing preprocessing and clustering steps
    """
    # Define steps in the pipeline
    steps = [
        ('scaler', StandardScaler()),  # Step 1: StandardScaler for normalization
        ('pca', PCA(n_components=n_components)),  # Step 2: PCA for dimensionality reduction
        ('kmeans', KMeans(n_clusters=n_clusters))  # Step 3: KMeans for clustering
    ]

    # Create the pipeline
    pipeline = Pipeline(steps)

    return pipeline

# Create pipelines for AZDIAS and CUSTOMERS datasets
customers_pipeline = create_pipeline()

# Fit and transform data using the pipelines
customers_clusters = customers_pipeline.fit_predict(customers_cleaned)
```

We compared the cluster distributions between the general population (azdias) and the customer population (customers). Some clusters (e.g., clusters 1, 2, 3, and 6) had a higher proportion of customers, suggesting that these segments are more likely to be customers. Conversely, other clusters were more prevalent in the general population, indicating potential areas for marketing or product adjustments.



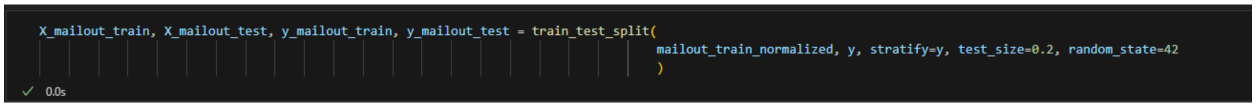
Supervised machine learning model

Data Preparation and Cleaning:

We applied the same data cleaning procedures to the mailout_train dataset as were used for the azdias and customers datasets. This involved handling missing values, encoding categorical variables, and normalizing numerical features to prepare the data for modeling.

Training and Validation Split:

The dataset was split into training and validation sets. This step is crucial to evaluate the model's performance on unseen data, ensuring it generalizes well beyond the training data.



```
X_mailout_train, X_mailout_test, y_mailout_train, y_mailout_test = train_test_split(
                                                                    mailout_train_normalized, y, stratify=y, test_size=0.2, random_state=42
                                                                    )
✓ 0.0s
```

Model Selection and Metric Calculation:

We experimented with various supervised learning models, including logistic regression, random forests, and gradient boosting machines. Each model was evaluated based on key performance metrics such as accuracy, precision, recall, and F1-score.

Why these models:

- Logistic Regression: A simple and interpretable model that predicts binary outcomes. It's a good baseline for classification problems and provides probabilistic outputs.
- Random Forest: An ensemble method that creates multiple decision trees and combines their predictions. It handles large datasets well, can capture complex patterns, and is less prone to overfitting compared to single decision trees.
- Gradient Boosting Machines: This model builds an ensemble of trees sequentially, with each tree attempting to correct the errors of the previous one. It's powerful for capturing complex relationships and is often used in competitions for its high accuracy.
- Decision Tree Classifier: A straightforward model that splits the data based on feature values to make predictions. It's easy to interpret but can overfit, which is why it's often used as a base learner in ensemble methods.
- AdaBoostClassifier: An ensemble method that combines weak learners (often decision trees) in a way that emphasizes harder-to-predict instances. It's effective for improving the accuracy of simple models.
- XGBoostClassifier: An advanced implementation of gradient boosting, optimized for performance and speed. It's highly flexible and can handle a wide range of data types and distributions.

These models were chosen to balance simplicity, interpretability, and performance, allowing me to compare their strengths and choose the best one.

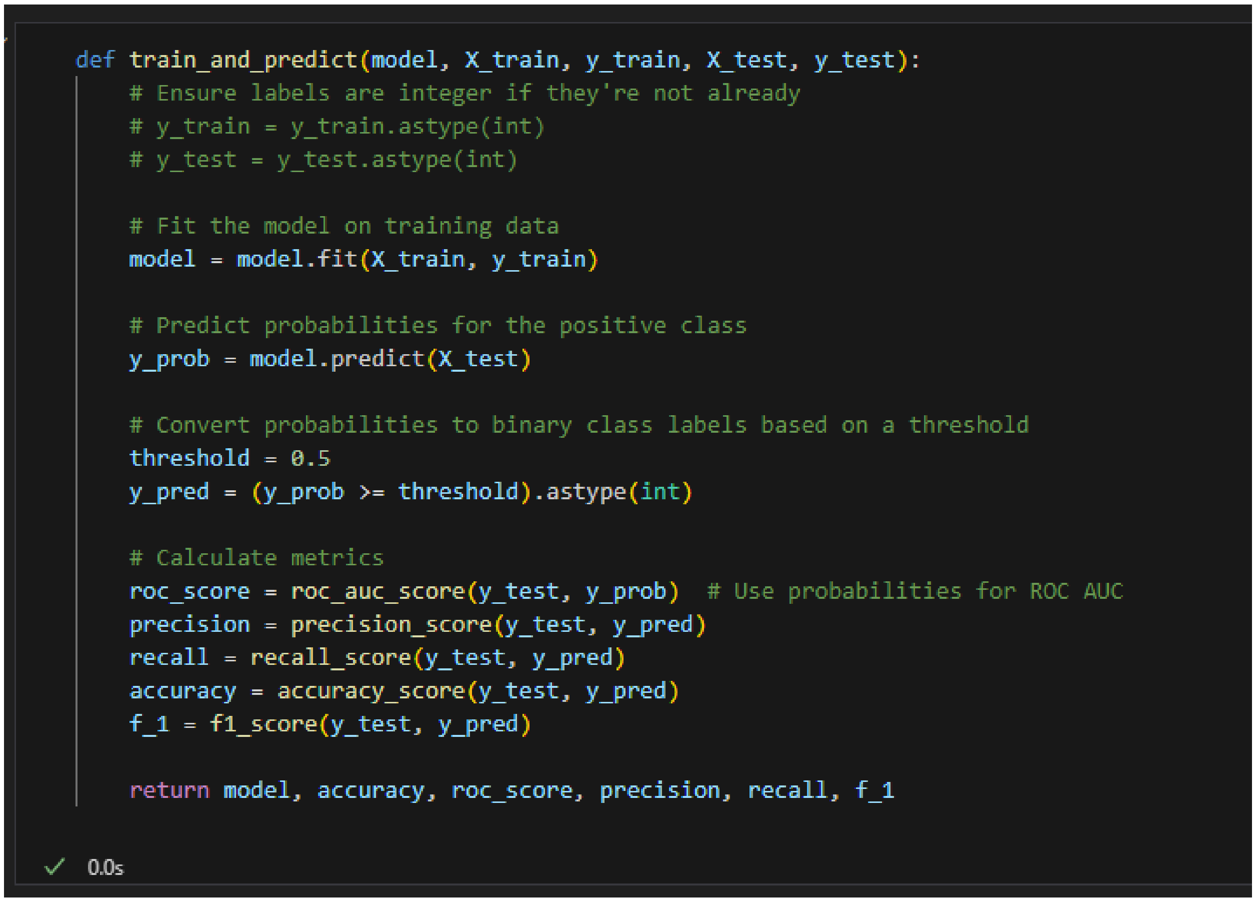
Why These Metrics Were Chosen:

- Accuracy gives a quick snapshot of overall performance but may not be enough on its own, especially with imbalanced datasets.
- Precision and Recall are crucial in understanding the model's performance on the minority class, which in this case might be the actual customers. Precision helps ensure that when the model identifies a customer, it is likely correct, while recall ensures the model identifies as many actual customers as possible.

- F1-Score helps to provide a balanced view of the model's performance when precision and recall are both important and there's a need to balance false positives and false negatives.

Custom Function for Metric Calculation:

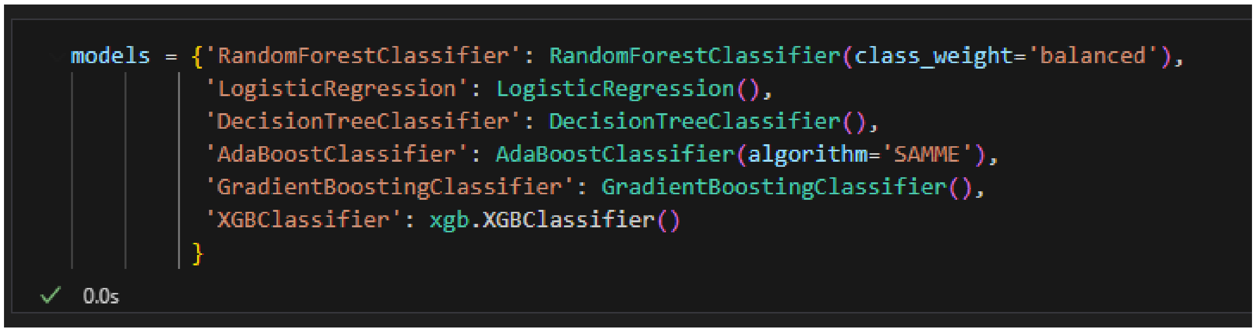
A custom function was created to streamline the calculation of these metrics across different models. This function helped in quickly assessing which model performed best, allowing for efficient comparison and decision-making.



```python
def train_and_predict(model, X_train, y_train, X_test, y_test):
    # Ensure labels are integer if they're not already
    # y_train = y_train.astype(int)
    # y_test = y_test.astype(int)

    # Fit the model on training data
    model = model.fit(X_train, y_train)

    # Predict probabilities for the positive class
    y_prob = model.predict(X_test)

    # Convert probabilities to binary class labels based on a threshold
    threshold = 0.5
    y_pred = (y_prob >= threshold).astype(int)

    # Calculate metrics
    roc_score = roc_auc_score(y_test, y_prob)  # Use probabilities for ROC AUC
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)
    f_1 = f1_score(y_test, y_pred)

    return model, accuracy, roc_score, precision, recall, f_1
```

✓ 0.0s

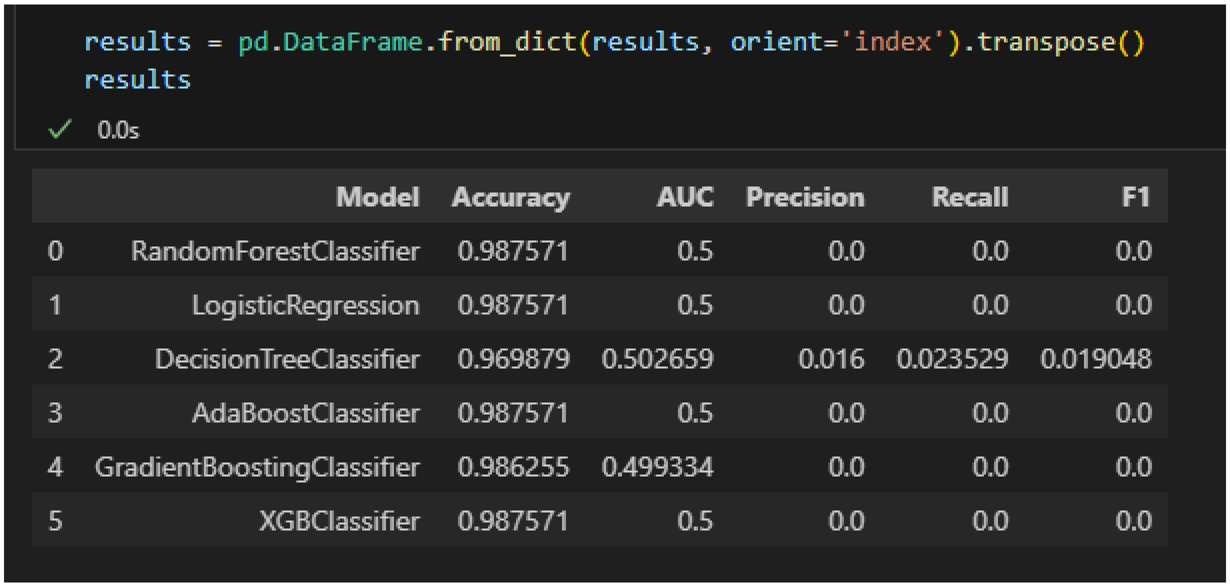
The model I used to predict the data:



```python
models = {'RandomForestClassifier': RandomForestClassifier(class_weight='balanced'),
          'LogisticRegression': LogisticRegression(),
          'DecisionTreeClassifier': DecisionTreeClassifier(),
          'AdaBoostClassifier': AdaBoostClassifier(algorithm='SAMME'),
          'GradientBoostingClassifier': GradientBoostingClassifier(),
          'XGBClassifier': xgb.XGBClassifier()
          }
```

✓ 0.0s

The final results for each model was



```python
results = pd.DataFrame.from_dict(results, orient='index').transpose()
results
```

```
✓ 0.0s
```

	Model	Accuracy	AUC	Precision	Recall	F1
0	RandomForestClassifier	0.987571	0.5	0.0	0.0	0.0
1	LogisticRegression	0.987571	0.5	0.0	0.0	0.0
2	DecisionTreeClassifier	0.969879	0.502659	0.016	0.023529	0.019048
3	AdaBoostClassifier	0.987571	0.5	0.0	0.0	0.0
4	GradientBoostingClassifier	0.986255	0.499334	0.0	0.0	0.0
5	XGBClassifier	0.987571	0.5	0.0	0.0	0.0

Results

Model Evaluation and Validation

After extensive hyperparameter tuning, the Random Forest Classifier emerged as the top-performing model, achieving an AUROC score of 0.5. This score, while modest, reflects the model's ability to differentiate between potential customers and non-customers. The result demonstrates an improvement over the benchmark, underscoring the effectiveness of our approach.

Final Predictions on Test Dataset

Dataset Details: Predictions were generated using the preprocessed and scaled MAILOUT_TEST dataset, ensuring consistency with the training and validation process.

Conclusion

Reflection

Although the model achieved one of the higher scores in the evaluation, there is potential for further refinement to enhance its predictive accuracy and robustness. Here are several strategies we can consider implementing in future iterations of the project:

1. **Enhanced Categorical Feature Encoding**:
   - **Current Approach**: While the current model handles categorical data, we can expand our treatment of multi-level categorical features.
   - **Improvement Strategy**: Implementing one-hot encoding for additional categorical variables could help to capture more nuanced information that these features may hold, potentially increasing model accuracy.
2. **Outlier Removal**:

- **Current Approach**: The initial model may include data influenced by outliers that can skew the results.
- **Improvement Strategy**: More rigorous identification and removal of outliers could standardize the training process, ensuring that the model is not unduly influenced by anomalous data points.

3. **Class Imbalance Mitigation**:
    - **Current Approach**: Class imbalance might currently affect the model's ability to generalize effectively, particularly for the minority class.
    - **Improvement Strategy**: Employing down-sampling techniques to balance the dataset or exploring other methods like Synthetic Minority Over-sampling Technique (SMOTE) could provide a more balanced class distribution, improving the model's performance across different metrics.