Week 5 - Feature engineering, Data augmentation, Dimensionality reduction

Aritra Ray
Maria Alice Fagundes Vieira

Feature Engineering

1. Label Encoding:

The categorical column `customer_code` was encoded using `LabelEncoder`. This converts each unique value in the `customer_code` column into a unique integer, allowing the model to interpret categorical data as numerical values, which is essential for machine learning algorithms.

2. Boolean Conversion:

The boolean columns were converted to integers (1 for True, 0 for False). This step standardizes the dataset, ensuring that all columns contain numerical values.

3. Feature Creation:

- Income to Product Ratio: This feature was created by dividing a customer's income by the number of products they hold (`total_products`). This ratio helps determine if higher-income customers tend to own more products.
- Income to Age Ratio: This metric was generated by dividing the income by the customer's age. It aims to capture disposable income, especially for younger or high-earning customers.
- Total Savings: A new feature `total_savings` was engineered by summing up various savings-related columns, such as `savings_acct`, `short_term_depo`, `medium_term_depo`, and `long_term_depo`. This feature provides an aggregate view of a customer's total savings across different accounts.

4. Feature Reduction:

Several non-numerical columns that are less relevant for modeling or highly categorical, such as `customer_code`, `employee_index`, `join_channel`, `province_name`, and `segment`, were dropped from the dataset to reduce complexity and prevent noise in the model.

Data Augmentation

Data augmentation typically refers to adding more features or transforming the data in a way that provides additional information to the model. We performed the following steps contribute to data augmentation:

- The creation of new features like `income_to_product_ratio`, `income_to_age`, and `total_savings` adds more dimensions for the model to learn customer behavior patterns.
- The `calculate_product_probabilities` function computes the likelihood of a customer using specific products. This probability-based information can be used to enhance the recommendation model by guiding it on which products are more popular and thus more likely to be recommended. However, these values are just for reference purposes, not to be augmented to the dataframe.

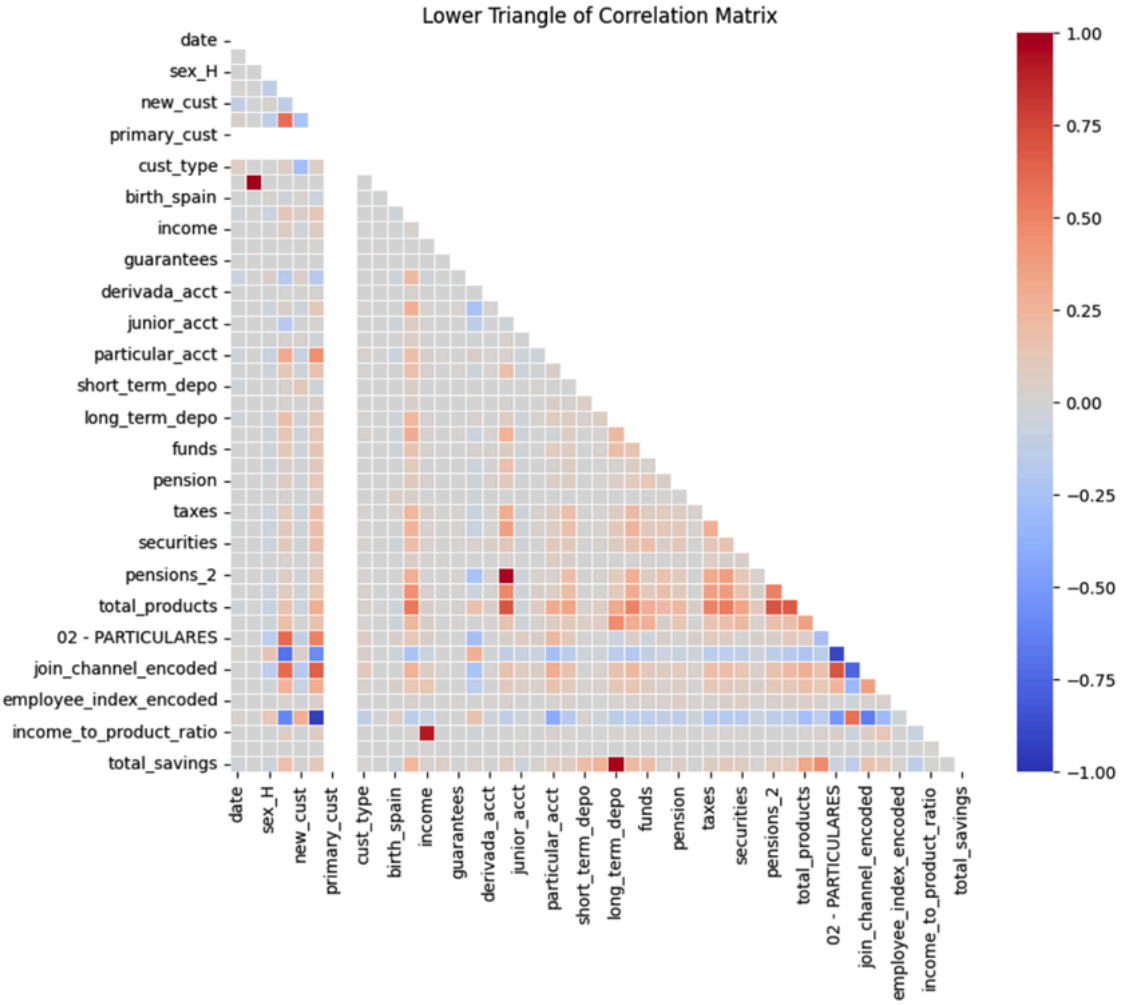
```
# Display the resulting DataFrame
print(probabilities)
```

```
                 Product  Probability
2          current_acct     0.618377
23          direct_debt     0.130428
7        particular_acct     0.126187
12               e_acct     0.085387
22          pensions_2      0.061872
4          payroll_acct     0.056650
21         payroll_acct     0.056650
17                taxes     0.055554
18          credit_card     0.045360
8   particular_plus_acct     0.042985
11       long_term_depo     0.042752
19           securities     0.025594
13                funds     0.018565
5           junior_acct     0.009486
15              pension     0.009366
6    mas_particular_acct     0.008211
14             mortgage     0.005934
20            home_acct     0.003935
16                loans     0.002406
10     medium_term_depo     0.001525
9       short_term_depo     0.001265
3         derivada_acct     0.000393
0          savings_acct     0.000105
1            guarantees     0.000024
```

Dimensionality Reduction

1. Correlation Analysis:

A correlation matrix was generated to evaluate how different features are related to each other. Since the features in the dataset were found to be only weakly correlated, applying Principal Component Analysis (PCA) for dimensionality reduction was deemed unnecessary in this case. The reason is that PCA works well when features are highly correlated, as it reduces redundancy by combining them into a smaller set of principal components. Here, PCA would not provide much benefit



2. Feature Selection:

A Random Forest Classifier was used to perform feature selection. After fitting the model, the feature importance scores were calculated, and only the features with importance above the

mean were selected. This step ensures that the model only focuses on the most relevant features, removing less significant ones to improve performance and reduce overfitting.
The selected features were stored in the `reduced_train` dataframe, which contains the reduced set of user-related features after selection.

Final Dataset

The dataset was ultimately divided into **product features** (related to the products offered) and **user features** (demographic and behavioral data). This separation is crucial for training a recommendation model that can predict which products each user is most likely to be interested in.

```
print("User Features Shape: ", user_features.shape)
print("Product Features Shape: ", product_features.shape)
```

```
User Features Shape:  (8224264, 23)
Product Features Shape:  (8224264, 24)
```