Report of Part 1: Traditional ML skills

Juliana Restrepo Trujillo

Contents

1. Problem statement

Customer retention is a critical focus for banks to ensure the longevity of their business. The objective is to analyze the customer data of account holders to predict and mitigate customer churn. The provided dataset contains information about the bank's account holders, and the aim is to develop a predictive model for customer churn.

You are required to perform data cleaning, an exploratory analysis, and build one or more classifiers, in order to predict the Customer Churn.

Shared files

● train.csv: the training dataset. Churn is the binary target to predict.
Note: Use this database to train and fine-tune the algorithm(s) that you choose. Also, use this dataset to evaluate your trained model and create a performance report.

● inference.csv - the inference dataset. Use this database in order to predict if the registers will be a Churn (1) or not (0).
Note: this file doesn't have the target column. It is going to be used for the prediction stage and evaluate your solution.

Guidelines:

1. Comments and code must be written in English.
2. Code should be commented according to Python docstrings.
3. Apply clean code and good programming practices.

4. Perform data exploration (variable description, possible values, missing values).
5. Perform data cleaning (if deemed necessary).
6. Conduct experiments with different models and parameters (use those deemed useful). Select a performance metric appropriate to the objective of the problem, and justify your choice. Also, consider secondary performance metrics you can consider when choosing the right algorithm.
7. Select the best model and explain your decision.
8. What would you do to improve the model in the future?
9. How do you imagine what you just did in OOP?
10. How do you deploy these models in production?
11. Upload the solution to a repo in https://github.com/ and share it with us!

2. Exploratory Data Analysis Report

2.1 Introduction

This report provides an exploratory data analysis (EDA) of the dataset provided. The main objectives are to describe the variables, identify possible values, and detect any missing values. This analysis is essential for understanding the dataset and preparing it for further modeling and analysis.

2.2. Data Description

The dataset consists of customer information from a bank. The target variable is Churn, which indicates whether a customer has left the bank. Table 1. Results of data description, presented below, shows the results related to each of the variables

Table 1. Data description results

Column (Variable)	**Description**	**Data Type visualized**
id	Unique identifier for each customer	Integer
CustomerId	Unique customer ID	Integer
Surname	Customer's surname	Object
CreditScore	Credit score of the customer	Integer
Geography	Customer's country of residence	Object
Gender	Customer's gender	Object
Age	Customer's age	Float
Tenure	Number of years the customer has been with the bank	Integer
Balance	Balance amount in the customer's account	Float
NumOfProducts	Number of products the customer has purchased	Integer
HasCrCard	Whether the customer has a credit card (1: Yes, 0: No)	Float
IsActiveMember	Whether the customer is an active member (1: Yes, 0: No)	Float
EstimatedSalary	Estimated annual salary of the customer	Float
Churn	Whether the customer has churned (1: Yes, 0: No)	Integer

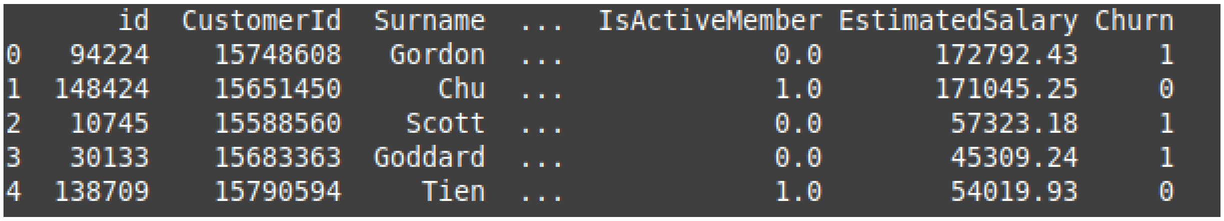
Source: own elaboration

2.3. Initial Data Exploration

First, it is verified that the database provided to the train.csv program is read correctly, then it is verified how the program reads each type of data. The results obtained are presented below in Figure 1 and 2.

Figure 1. First 5 Rows of the Dataset, shows the result obtained by running the code data1043.py. The purpose of printing the first 5 rows of the train.csv file is to see if it is read correctly.

Figure 1. First 5 Rows of the Dataset



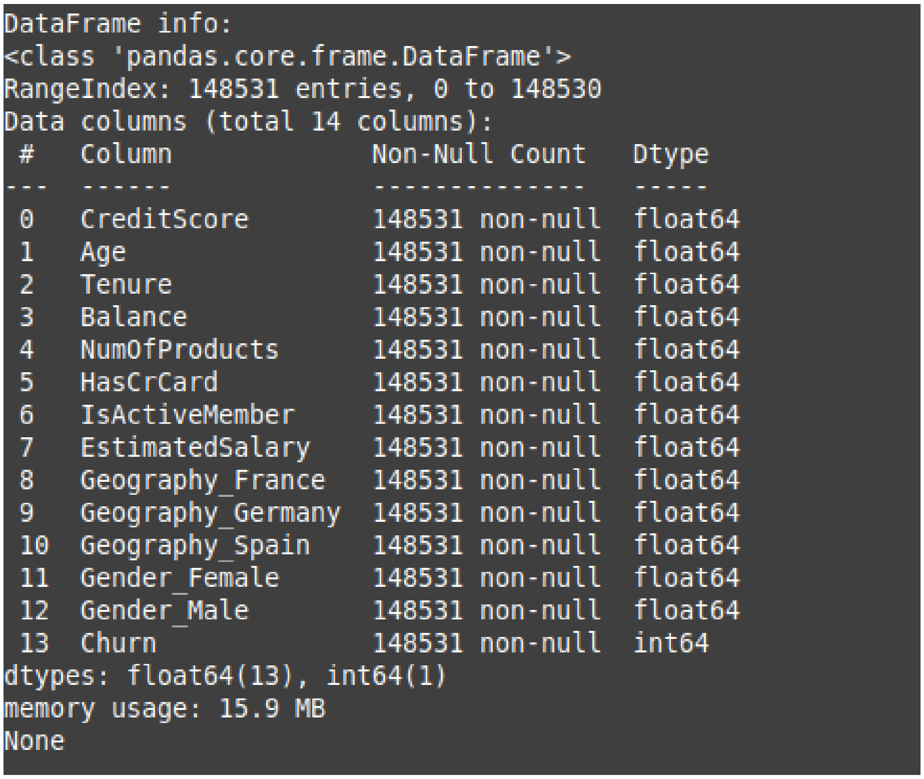
```
       id  CustomerId  Surname  ...  IsActiveMember EstimatedSalary Churn
0   94224    15748608   Gordon  ...             0.0       172792.43     1
1  148424    15651450      Chu  ...             1.0       171045.25     0
2   10745    15588560    Scott  ...             0.0        57323.18     1
3   30133    15683363  Goddard  ...             0.0        45309.24     1
4  138709    15790594     Tien  ...             1.0        54019.93     0
```

Source: own elaboration

It can be seen that the file is read correctly by the python.

Figure 2. Data Frame info , shows the datatype proporcinated for the program.

Figure 2. Data Frame info



```
DataFrame info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148531 entries, 0 to 148530
Data columns (total 14 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   CreditScore        148531 non-null  float64
 1   Age                148531 non-null  float64
 2   Tenure             148531 non-null  float64
 3   Balance            148531 non-null  float64
 4   NumOfProducts      148531 non-null  float64
 5   HasCrCard          148531 non-null  float64
 6   IsActiveMember     148531 non-null  float64
 7   EstimatedSalary    148531 non-null  float64
 8   Geography_France   148531 non-null  float64
 9   Geography_Germany  148531 non-null  float64
 10  Geography_Spain    148531 non-null  float64
 11  Gender_Female      148531 non-null  float64
 12  Gender_Male        148531 non-null  float64
 13  Churn              148531 non-null  int64
dtypes: float64(13), int64(1)
memory usage: 15.9 MB
None
```

Source: own elaboration

It is concluded that the information related to the type of variable observed differs from that of the programme.

2.4. Data Cleaning

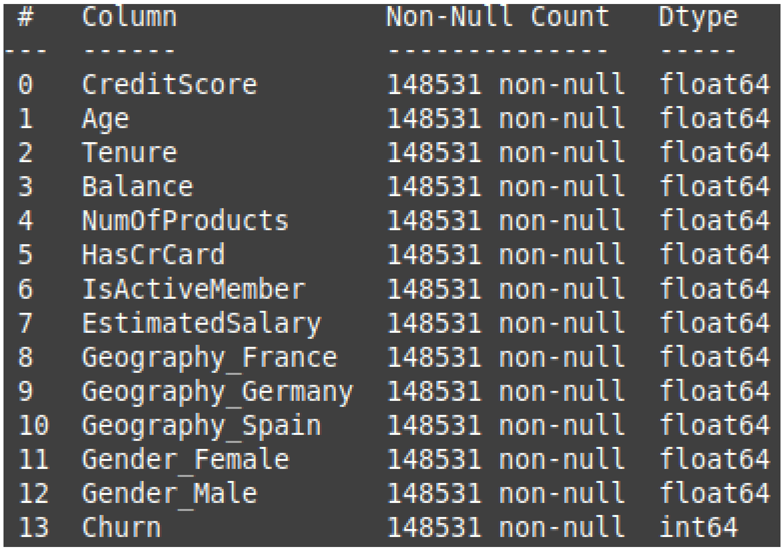
2.4.1 Drop Irrelevant Columns

The columns for the variables id, Customer Is, Surname were removed because the program did not want to modify them.

2.4.2 Handle Missing Values and Encode Categorical Features

The main purpose of this step was to change the Data type recognized by the program to the Data type observed by me in the database provided. In Figure 3. Data type. It can be observed that the data type identified by the program is the same as the one observed in the database.

Figura 3. Data type



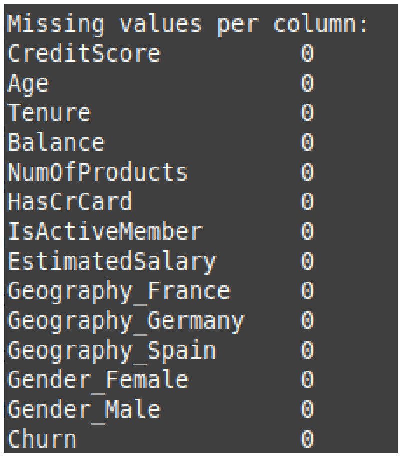
```
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   CreditScore        148531 non-null  float64
 1   Age                148531 non-null  float64
 2   Tenure             148531 non-null  float64
 3   Balance            148531 non-null  float64
 4   NumOfProducts      148531 non-null  float64
 5   HasCrCard          148531 non-null  float64
 6   IsActiveMember     148531 non-null  float64
 7   EstimatedSalary    148531 non-null  float64
 8   Geography_France   148531 non-null  float64
 9   Geography_Germany  148531 non-null  float64
 10  Geography_Spain    148531 non-null  float64
 11  Gender_Female      148531 non-null  float64
 12  Gender_Male        148531 non-null  float64
 13  Churn              148531 non-null  int64
```

Source: own elaboration

2.4.3 Verify the Cleaning

The data were checked for cleanliness by checking the database for missing data. It should be noted that typographical errors present in the database in the Surname column were not taken into account for two main reasons: 1) they are errors due to the use of accents; and 2) they did not influence the purpose of this study, to make a model to predict customer prediction. In Figure 4. Missing values per column, it is observed the verification

Figure 4. Missing values per column



```
Missing values per column:
CreditScore          0
Age                  0
Tenure               0
Balance              0
NumOfProducts        0
HasCrCard            0
IsActiveMember       0
EstimatedSalary      0
Geography_France     0
Geography_Germany    0
Geography_Spain      0
Gender_Female        0
Gender_Male          0
Churn                0
```

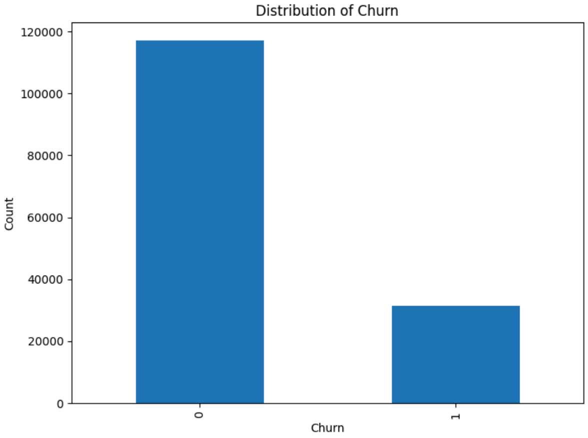
Source: own elaboration

2.5. Exploratory Data Analysis (EDA)

2.5.1 Distribution of Target Variable

The purpose of this point is to better understand the nature of the variable we are trying to predict. This part shows the distribution of the variable Churn, Figure 5. Distribution of Churn.

Figure 5. Distribution of Churn

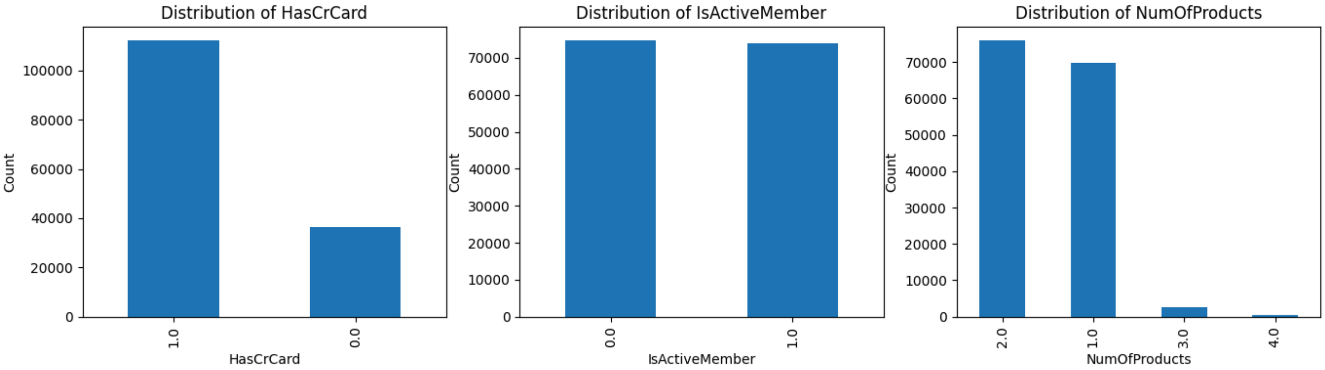


Source: own elaboration

2.5.2 Distribution of Binary Variables

This item visualises the graphs of the binomial variable distributions in order to better understand the nature of these variables and their potential impact on the target variable (Churn). This was done in order to Understand the Distribution of Variables, Identify Patterns and Trends of variables, pre-select characteristics, visualise and communicate results and This analysis also guides the selection and transformation of characteristics, and for the identification of the most relevant characteristics for the elaboration of the predictive model. Figure 6. Distributions of Binary Variables. The distribution plots are presented.

Figura 6. Distributions HasaCard, IsActiveMember and NumofProducts. Se presntan las gráficas de distribución.

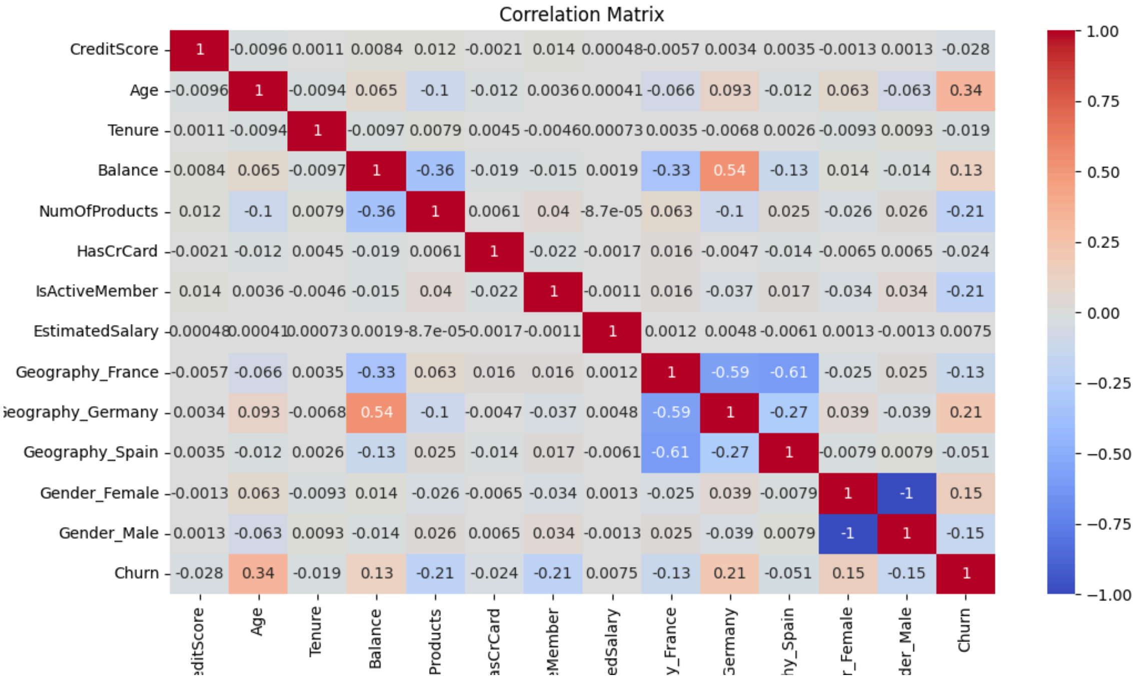


Source: own elaboration

2.5.3 Correlation Matrix

The correlation matrix is a fundamental statistical tool for identifying and understanding relationships between numerical variables, selecting variables for predictive model development, and verifying initial hypotheses. This analysis is crucial to prepare the data for predictive modelling and to better understand the dynamics between the study variables. Figure 7. Correlation Matrix, shows the results obtained with the data code.

Figure 7. Correlation Matrix



Source: own elaboration

2.5.4 Analysis of the results obtained in the correlation matrix related to Churn.

Between the variables Age and Churn there is a moderate positive correlation between age and churn. This suggests that older customers are more likely to leave the bank. This could be because older customers are less likely to take up new banking products or services.

The correlation between the variables Number of products and Churn is moderately positive. This indicates that customers with more products are more prone to churn. This could be contradictory and warrants further research to understand the underlying reasons.

Las variables Churn y IsActiveMember son inversas lo que significa que tienen menos probabilidades de abandonar el banco.

There is a weak negative correlation between Balance and Churn. Customers with higher balances are slightly less likely to churn. This makes sense, as customers with higher balances may have invested more in staying with the bank.

Germany Customers have a weak positive correlation with Churn, suggesting that German customers are slightly more likely to leave the bank.

Women are slightly more likely to Churn than men.

The variables Credit score, Seniority, Estimated salary have very weak correlations with churn, suggesting that they are not significant factors in the probability of Churn.

2.5.5 Conclusion

The exploratory data analysis provided insights into the dataset. We observed the distributions of key variables and identified that there are no missing values in the dataset. The correlation matrix revealed relationships between numerical variables, which will be useful for further analysis and modeling. Notably, the positive correlation between NumOfProducts and Churn warrants further investigation to understand the underlying reasons. The moderate correlation between Age and Churn suggests that older customers are more likely to churn, which could inform targeted retention strategies.

3. Report of model selection

3.1. Experiments with Different Models and Parameters

3.1.1. Description of the Tested Models

To address the Churn prediction problem, the following machine learning models Logistic Regression, Decision Tree and Random Forest Classifier were tested. Following will explain each one.

- Logistic Regression: Logistic regression is a simple and highly interpretable model that provides probabilities and coefficients that can be easily understood. This is especially useful in problems where model interpretability is crucial. It is a good starting point for binary classification problems and is useful for establishing a performance baseline.

- Decision Tree: Decision trees can capture non-linear relationships between features and the target variable, which is beneficial when the relationships are non-linear.

Additionally, it can handle both categorical and numerical variables without the need for special preprocessing and is capable of capturing complex interactions between variables.

- Random Forest Classifier: is an ensemble method that creates multiple decision trees and combines their predictions to improve accuracy and reduce overfitting because it uses averages of multiple decision trees to avoid overfitting and improve performance on unseen data.

3.1.2. Evaluation Metrics

The primary and secondary metrics selected are:

- Primary Metric: AUC (due to the importance of the model's ability to distinguish between positive and negative classes in an imbalanced binary classification problem).

- Secondary Metrics:
    - ◦ Precision: To measure the accuracy of positive predictions.
    - ◦ Recall: To evaluate the model's ability to identify all positive instances.
    - ◦ F1-Score: To balance precision and recall.
    - ◦ Accuracy: To have an overview of performance.

The results obtained by the models in the proposed metrics are presented in Table 2. Results of models analyzed.

Table 2. Results of models analyzed

Model	AUC	Precisión	Recall	F1-Score	Accuracy
Logistic Regression	0.84	0.70	0.40	0.51	0.84
Random Forest Classifier	0.86	0.77	0.49	0.60	0.86
Decision Tree	0.71	0.77	0.48	0.56	0.86

Source: own elaboration

3.2. Model section

3.2.1. Decision criteria

The best model was selected based on the following considerations:

- AUC: Main evaluation metric due to its ability to handle class imbalance and its clear interpretation in terms of classification capacity.
- F1-Score: Important to balance precision and recall, especially in a problem where both metrics are critical.
- Overall Performance: Precision, recall, and accuracy were also considered for a comprehensive view of the model's performance.

3.2.2. Performance of the selected model

The Random Forest Classifier model was selected as the best model based on its superior performance on the AUC metric and a good balance in precision, recall and F1-Score.

The results of the metrics are presented in Table 3. Results of the metrics of the selected model

Table 3. Results of the metrics of the selected model

Metric	Value
AUC	0.86
Precisión	0.77
Recall	0.49
F1-Score	0.60
Accuracy	0.86

Source: own elaboration

3.2.3. Selection Justification

The Random Forest Classifier model showed the best overall performance, with the highest AUC score and a good balance between precision and recall, suggesting that it is able to effectively distinguish between churning and non-churning customers. Furthermore, the model had high precision and a good F1-Score, indicating that it is a robust choice for this classification problem.

4. Suggestions for improvement of the proposed model

To improve the customer churn prediction model, several strategies could be considered based on the analysis of the results obtained and the best practices in the field of machine learning. Below are some ways in which the proposed prediction model could be improved.

4.1. Create New Features

Create features that represent interactions between important variables, add time-based features, apply logarithmic or polynomial transformations to capture non-linear relationships.

4.2. Data Collection and Enrichment

- **External Data Sources** : Integrate relevant external data, such as economic indicators, additional credit information, and demographic data.
- **Behavioral Data** : Include more data about customer behavior, such as detailed transaction history, customer service interactions, and website or mobile application usage patterns.

4.3. Managing Class Imbalance

- **Resampling Techniques** :
  - **SMOTE (Synthetic Minority Over-sampling Technique)** : Generate new synthetic samples for the minority class.
  - **Undersampling** : Reducing the number of samples from the majority class to balance the classes.
- **Class Weight Adjustment** : Modify the class weights in the model to give more importance to the minority class.

4.4. Model Optimization

- **Hyperparameters** :
  - **Exhaustive Search (Grid Search)** or **Random Search** : Perform a more exhaustive search for hyperparameters.

  - **Bayesian Optimization** : Use advanced methods such as Bayesian optimization to find the best hyperparameters.
- **Advanced Models** :
  - **XGBoost, LightGBM, CatBoost** : Test advanced boosting models that often outperform random forests on classification tasks.

4.5. Model Assemblies

- **Stacking** : Combining multiple base models to form a meta-model.
- **Blending** : Trying different combinations of models and assembly methods to improve performance.

4.6. Validation and Evaluation

- **Cross Validation** : Use k-fold cross-validation to evaluate model performance more robustly.
- **Learning Curves** : Analyze learning curves to identify overfitting or underfitting problems.

4.7. Interpretability and Explainability

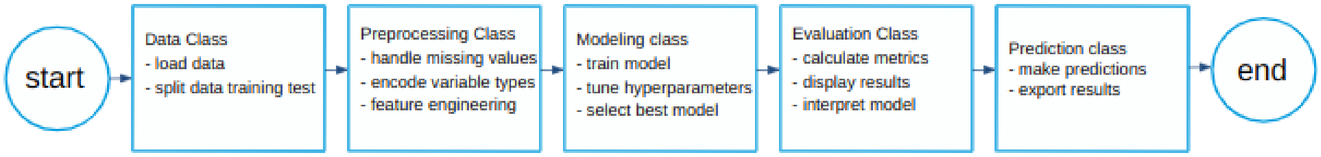
- **SHAP (SHapley Additive exPlanations)** : Use SHAP to interpret model predictions and better understand the importance of each feature.
- **LIME (Local Interpretable Model-agnostic Explanations)** : Use LIME to provide local explanations of predictions.

5. How do you imagine what you just did in OOP?

Implementing customer churn analysis using an object-oriented programming (OOP) approach involves creating several classes that encapsulate different aspects of the machine learning process. In this way, the basic principles of OOP (Classes and Objects, Encapsulation, Inheritance, Polymorphism, and Abstraction) will be taken into account.

The proposed classes and methods for this approach are presented below: the first class corresponds to Data Class, in this phase data is loaded into the program and cleaned; in the second, identified as Preprocessing Class, handle missing values, encode variable types, and feature engineering are performed; in the third, Modeling Class, model is trained, hyperparameters are tuned, and best model is selected; in the fourth, Evaluation Class, metrics are calculated, results are displayed, and model is interpreted; and in the fifth, Prediction Class, predictions are made and results are exported. This process is shown in Figure 8. Flowchart of the process of modeling customer churn in a bank.

Figure 8. Flowchart of the process of modeling customer churn in a bank



Source: own elaboration

In addition to changing the type of development approach, it is important to take into account the difference in results of the different types of approaches. The procedural approach is quick and easy to implement, however, code maintenance becomes complex as more models, features, and analysis are added. On the other hand, the OOP approach requires more time for initial planning and structuring of the code, but the code is easy to maintain and scale. As for the model performance metrics, they are maintained.

For large, long-term projects, the OOP approach is more suitable. For small projects or quick analyses, the procedural approach is better.

6. How do you deploy these models in production?

Deploying machine learning models in production involves several steps, including preparing the model, setting up the infrastructure, and ensuring the model can be accessed and used reliably. The basic steps to implement the model are:

6.1. Model Preparation.

- **Training**: Train the model using historical data and save the trained model.
- **Serialization**: Serialize the model using libraries like pickle or joblib in Python. This step ensures that the model can be saved and loaded later.
- **Validation**: Validate the model to ensure it meets performance criteria and is ready for deployment.

6.2. Setting Up the Infrastructure

- **Environment**: Ensure that the environment where the model will be deployed matches the environment where it was trained. This includes matching software versions, dependencies, and libraries.
- **Containerization**: Use Docker to containerize the model and its dependencies. This makes the deployment process more consistent and scalable.

6.3. Deployment Methods

- **REST API**: Deploy the model as a REST API using frameworks like Flask, FastAPI, or Django.

- **Serverless Functions**: Use cloud provider serverless offerings like AWS Lambda, Google Cloud Functions, or Azure Functions.
- **Batch Processing**: For use cases that do not require real-time predictions, deploy the model as part of a batch processing pipeline using tools like Apache Spark.

6.4. Monitoring and Maintenance

- **Logging and Monitoring**: Implement logging and monitoring to track the performance and usage of the model in production. Use tools like Prometheus, Grafana, or cloud-specific monitoring services.
- **A/B Testing**: Deploy multiple versions of the model and perform A/B testing to compare their performance.
- **Model Retraining**: Set up a pipeline for regular retraining of the model with new data to ensure it stays up to date.