MINDD – 1st Practical Work

António azevedo - 1250494

Diogo Silva - 1211882

Hugo Ribeiro - 1250520

INDEX

Data Mining Goals

The goal of this project is to build data-driven models that predict the success of telemarketing calls aimed at selling long-term bank deposits. Data mining extracts actionable patterns from large datasets; in this context, it can reveal customer and campaign factors that meaningfully improve business decisions.

In our dataset, the target variable is known as *"y"* (categorical: *yes* or *no*), which indicates whether or not a client subscribed to a long-term bank deposit during the current campaign.

Dataset

Bank Client Data

1. age — Age of the client (numeric)
2. job — Type of job (categorical)
3. marital — Marital status (categorical)
4. education — Level of education (categorical)
5. default — Has credit in default? (categorical)
6. housing — Has a housing loan? (categorical)
7. loan — Has a personal loan? (categorical)

Related to the Last Contact of the Current Campaign

8. contact — Contact communication type (categorical)
9. month — Last contact month of the year (categorical)
10. day_of_week — Last contact day of the week (categorical)
11. duration* — Last contact duration in seconds (numeric)

***Important Note:**

The *duration* attribute strongly influences the output (for example, if *duration* = *0*, then *y* = *"no"*). However, *duration* is not known before making the call, and the output *y* is only known afterwards. Therefore, *duration* was used only for benchmarking purposes and excluded from the realistic predictive models.

Other Attributes

12. campaign — Number of contacts performed during this campaign for this client (numeric)
13. pdays — Number of days since the client was last contacted in a previous campaign (numeric)
14. previous — Number of contacts performed before this campaign for this client (numeric)
15. poutcome — Outcome of the previous marketing campaign (categorical)

Social and Economic Context Attributes

16. emp.var.rate — Employment variation rate (quarterly indicator, numeric)
17. cons.price.idx — Consumer price index (monthly indicator, numeric)
18. cons.conf.idx — Consumer confidence index (monthly indicator, numeric)
19. euribor3m — Euribor 3-month rate (daily indicator, numeric)
20. nr.employed — Number of employees (quarterly indicator, numeric)

Target Variable (Output)

21. y — Has the client subscribed to a term deposit? (binary: *yes*, *no*)

Success Criteria

Project success is defined by the model's ability to accurately predict whether a client will subscribe to a term deposit. Performance was evaluated using classification metrics such as ROC-AUC, F1-score, and precision-recall.

Exploratory Analysis

The dataset exploration began with examining its size and structure:

- Number of rows and columns: 41,188 × 21
- Total elements: 864,948

An overview of both numerical and categorical variables was conducted to identify missing values and their percentages per column. The analysis revealed that numerical columns contained no missing values, while several categorical columns did:

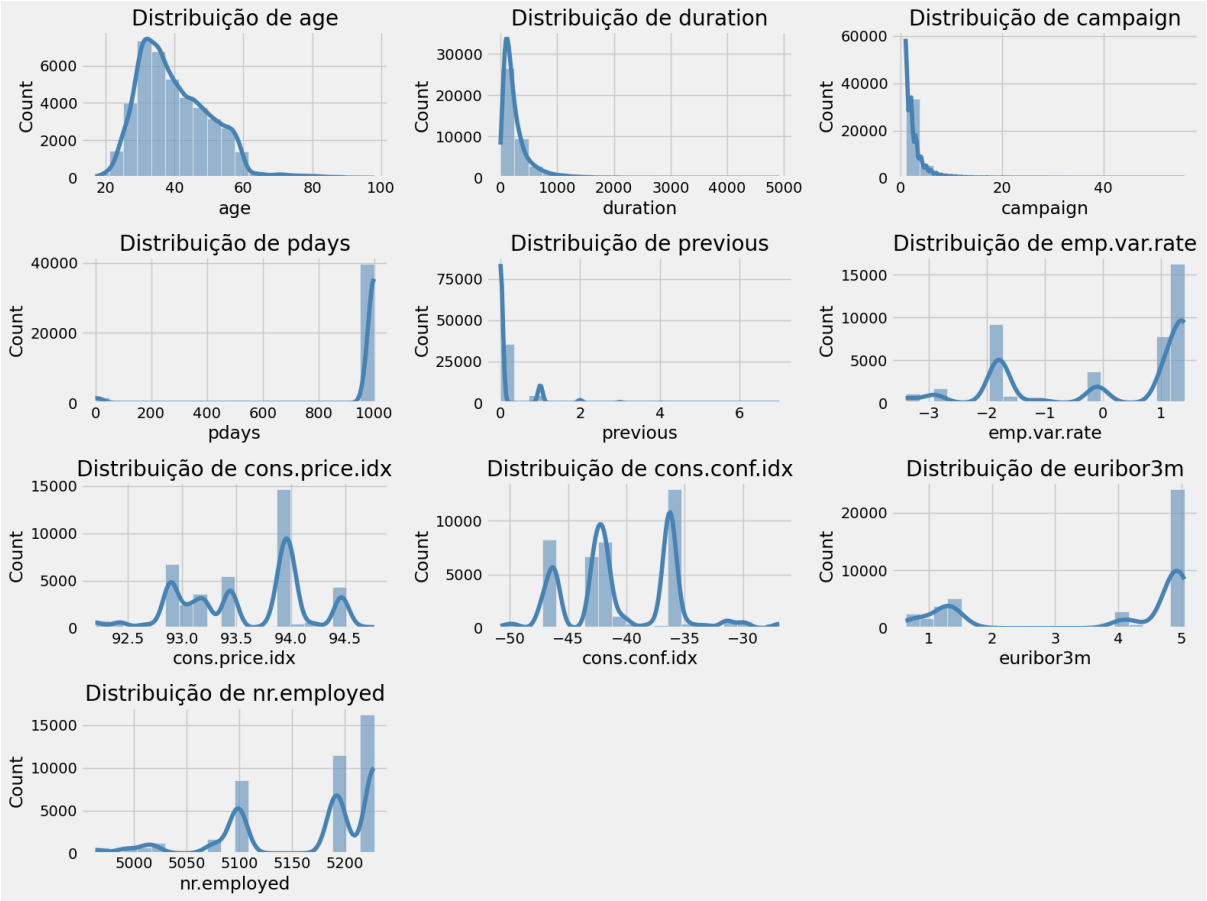
Variable	Missing Values (%)
default	20.87%
education	4.20%
housing	2.40%
loan	2.40%

job	0.80%
marital	0.19%

Numerical Feature Distribution

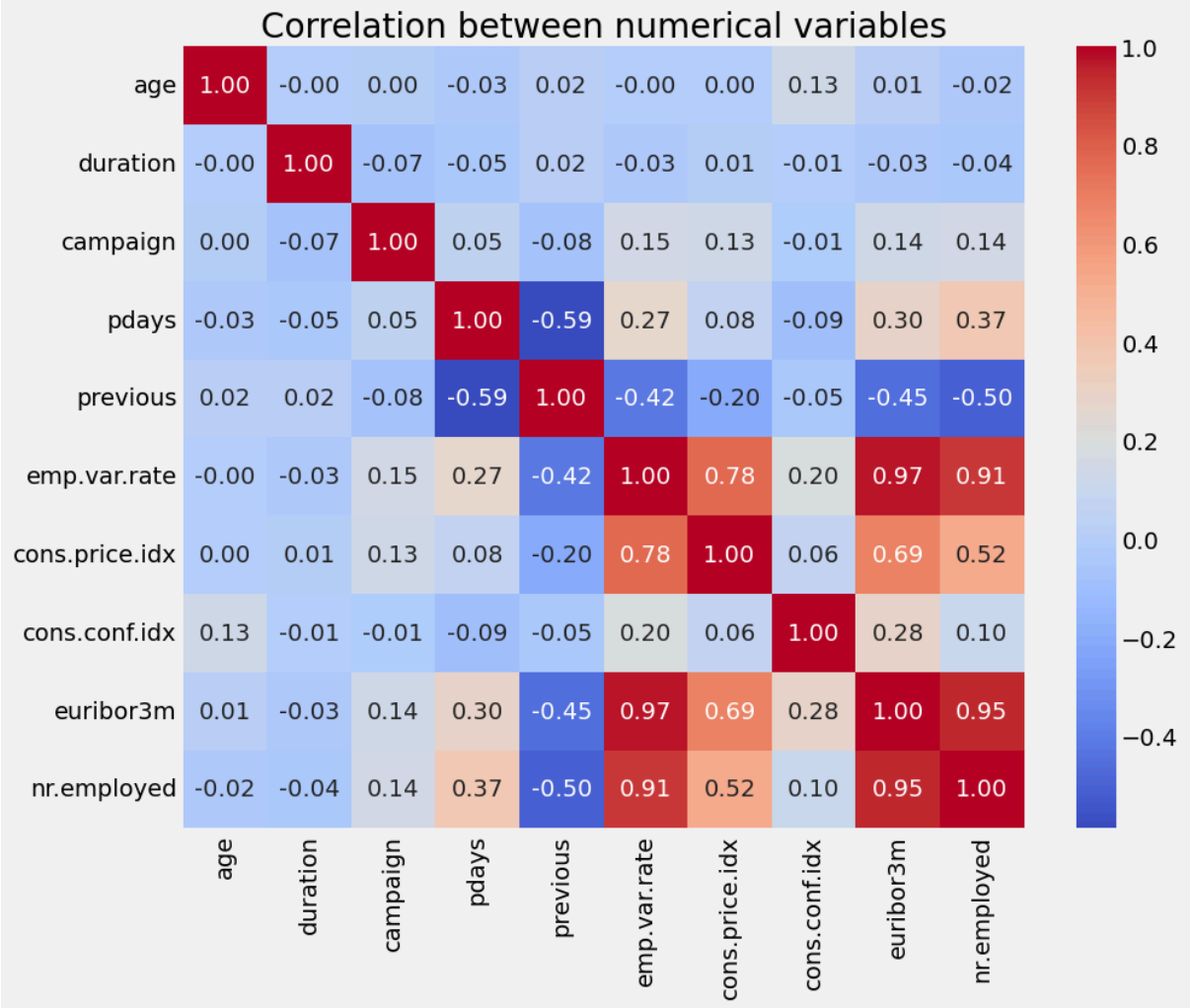
An analysis of the distribution of numerical features was performed, allowing us to better understand the underlying patterns and tendencies present within the dataset. From this analysis, we can conclude that:

- age – right-skewed suggesting adult clients predominate between 30 and 40 years old.
- duration, campaign, previous – strongly right-skewed suggesting many short calls or few contacts.
- pdays – displays a bimodal pattern, with a pronounced peak at 999.

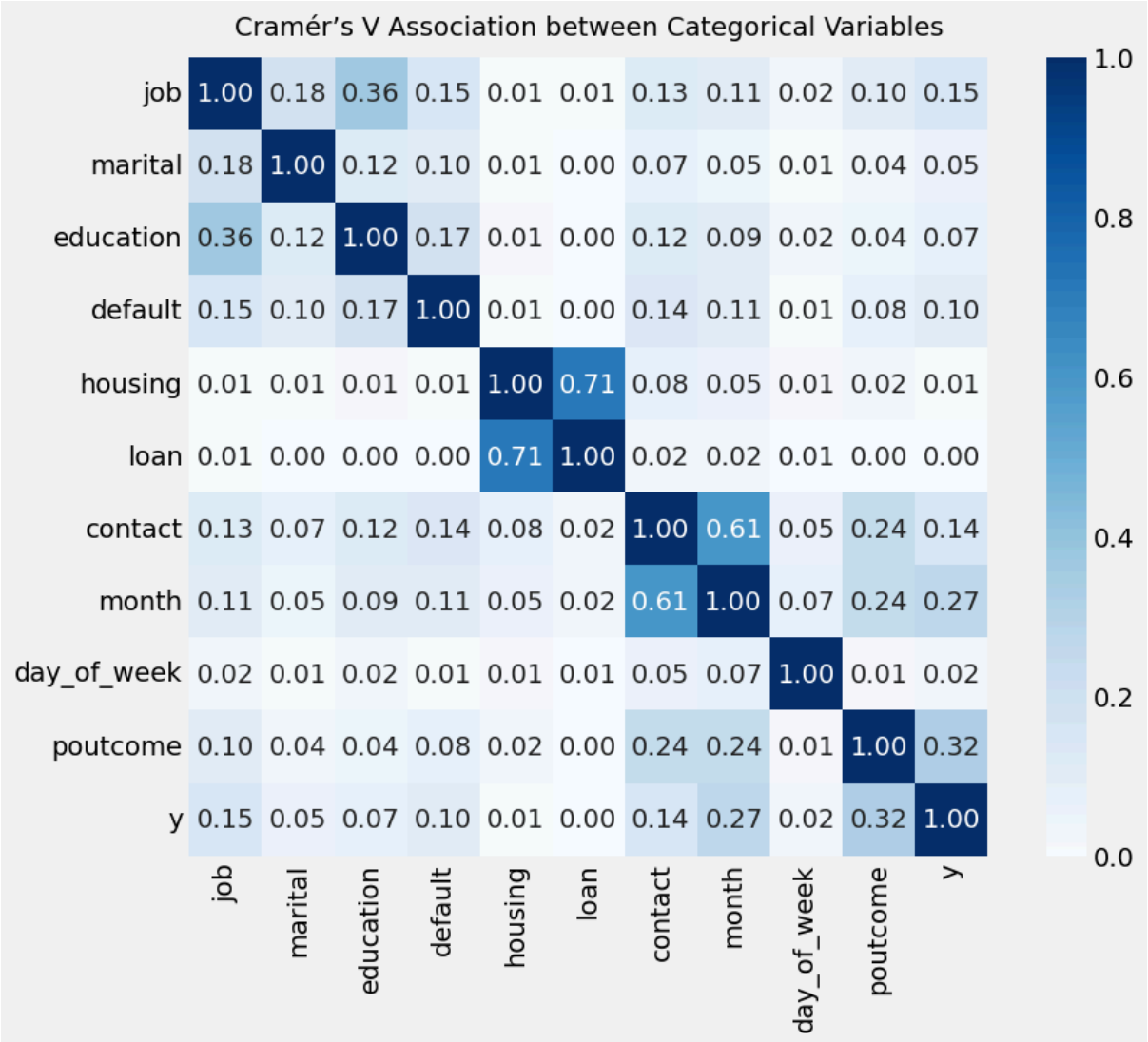


Variable Correlation

Correlations between variables were visualized through heatmaps for both numerical and categorical data.



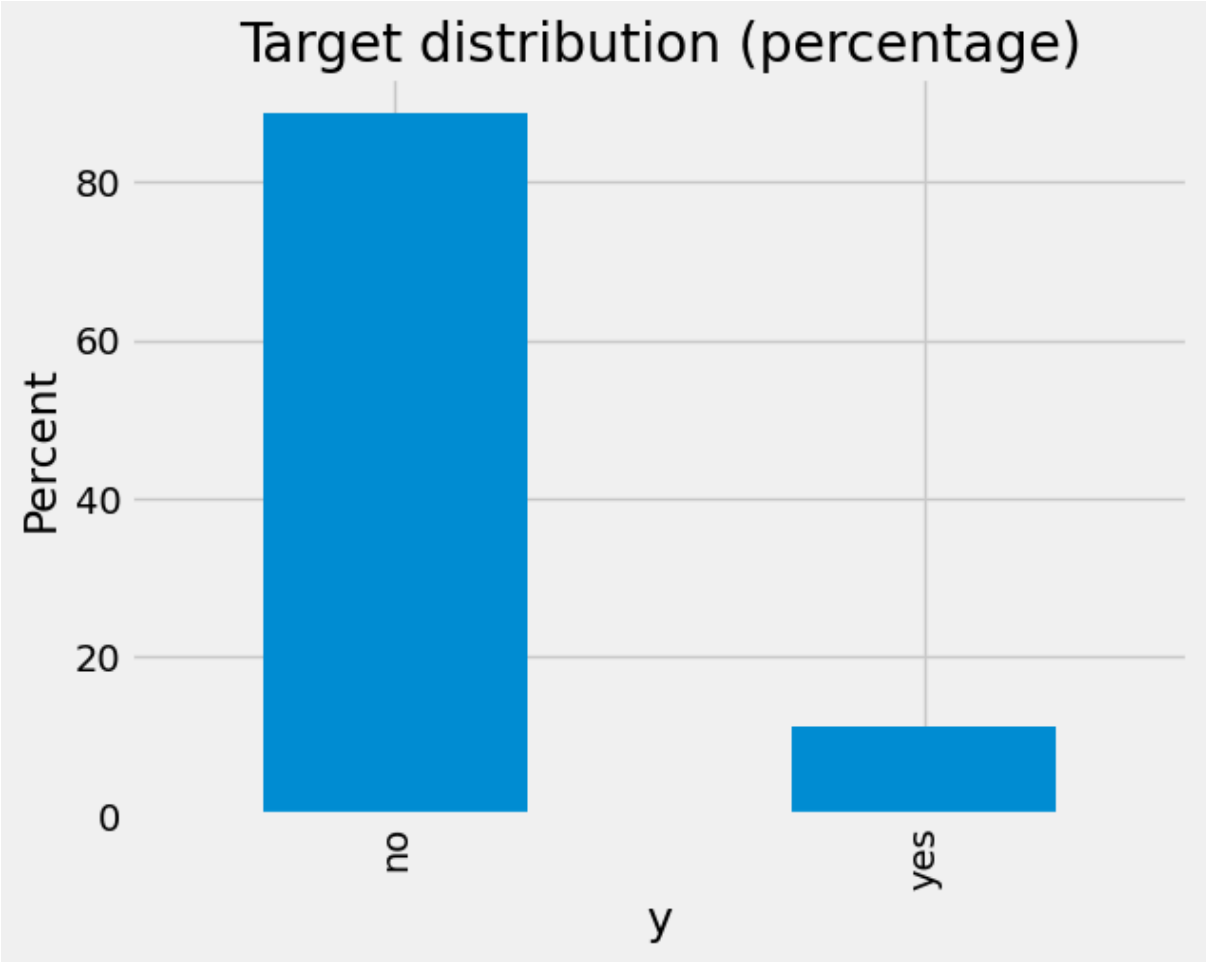
After analyzing the correlations among the numerical variables, a clear case of multicollinearity was observed between **emp.var.rate**, **euribor3m** and **nr.employed** as all three showed strong intercorrelations with values exceeding **0.80**. This indicates that these variables capture very similar economic information and may introduce redundancy in predictive modeling.



In contrast, the correlations among the categorical variables and the target variable were generally moderate to low. The highest correlation values were observed for **loan (0.71)** and **month (0.61)**. Since none of these exceed the **0.80** threshold, no significant multicollinearity was detected among the categorical variables, indicating that they contribute relatively independent information to the model.

Class balance plot

The distribution of the target variable (***y***) was analyzed to determine the proportion of clients who subscribed *("yes")* versus those who did not (*"no"*). Through this analysis, we observed that the target variable is highly imbalanced, with only about **11% *"yes"*** responses. This imbalance indicates that the dataset is dominated by negative cases, which should be addressed during model training to avoid bias toward the majority class.



Data Cleaning & Preprocessing

Before modeling, the dataset was subjected to a thorough cleaning and preprocessing phase to ensure data integrity and analytical consistency.

To begin, three variables **cons.price.idx**, **nr.employed** and **euribor3m** were removed due to strong multicollinearity (correlation coefficients above **0.80**). This step helped prevent redundancy and improved the reliability of the predictive models.

No true null values were found in the dataset; however, categorical variables containing *“unknown”* entries were carefully reviewed. In some cases, such as **default** and **education** their *“unknown”* values were retained as meaningful categories, while in others, they were removed when representing less than 3% of total observations. Duplicate records were also identified and eliminated to maintain data uniqueness.

Categorical variables were then encoded appropriately: binary attributes were transformed using ***LabelEncoder***, whereas multi-class features (e.g., job, marital, education, contact, month, poutcome) were processed using ***OneHotEncoder***. Numerical variables were standardized using ***StandardScaler*** to ensure that models sensitive to feature magnitudes, such as ***Logistic Regression*** and ***KNN***, performed optimally.

Train/Test Split and Cross-Validation

To evaluate model performance, the dataset was divided into training and testing subsets. After experimenting with multiple configurations (70/30, 80/20, and 90/10) an **80/20 split** was selected. This configuration was chosen because it provided the best results during preliminary testing.

Additionally, to ensure that the model's performance was not dependent on a single random partition, a ***Stratified K-Fold Cross-Validation*** approach was applied. This method preserves the class distribution within each fold and averages performance across multiple train/test splits, resulting in a more stable and generalizable estimate of predictive capability.

Following this setup, the ***SMOTE (Synthetic Minority Oversampling Technique)*** method was applied to the training data to address the class imbalance observed in the target variable (**y**), ensuring that the minority class was adequately represented during model learning.

In summary, these preprocessing steps resulted in a clean, consistent, and well-structured dataset suitable for binary classification modeling.

Modelling

A dedicated evaluation function was developed to systematically assess the performance of each classifier across multiple **classification thresholds** (0.3, 0.5, and 0.7).
This approach provides a comprehensive understanding of how variations in the decision boundary affect the model's **sensitivity** (recall) and **precision**.

For each model, the classifier is first trained on the training subset. The predicted probabilities, obtained through the predict_proba() method, are then used to compute the **ROC-AUC** and generate the **ROC Curve**, illustrating the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR).

For every threshold value tested:

- Predicted classes are derived directly from the probability estimates (y_proba ≥ threshold).

- Key performance metrics are calculated, including ***Precision***, ***Recall***, ***F1-Score***, ***Balanced Accuracy***, and the ***Matthews Correlation Coefficient (MCC)***.

- The ***Confusion Matrix*** is plotted to visualize classification outcomes, distinguishing correctly and incorrectly classified instances (True Positives, False Positives, False Negatives, and True Negatives).

This unified evaluation framework ensures a **consistent and objective comparison** between different classifiers, providing insights into both overall discriminative power (AUC) and threshold-dependent classification behavior.
The analysis was applied to multiple algorithms, enabling an informed selection of the most robust and balanced predictive model.

Random Forest

The ***Random Forest*** algorithm is an ***ensemble model*** that combines multiple decision trees to improve prediction accuracy and robustness. By averaging the predictions of several trees trained on different subsets of data and features, it reduces variance and minimizes overfitting compared to a single tree.

The model was trained with default parameters and evaluated through both test set analysis and cross-validation. This approach provided a balance between predictive power and model stability.

Adjusted parameters:

- **random_state = 42** — guarantees reproducibility of results.

Logistic Regression

The ***Logistic Regression*** model was selected as a ***baseline classifier*** for its simplicity, interpretability, and efficiency. It estimates the probability of class membership using a logistic function, providing a strong reference point for comparing more complex models.

This model is particularly suitable for linearly separable data and offers straightforward interpretation of feature influence on the target variable. Its probabilistic output enables flexible threshold adjustment to optimize recall or precision depending on the classification goal.

Adjusted parameters:

- **max_iter = 1000** — increases the maximum number of optimization iterations to ensure convergence during model training on larger datasets.

Decision Tree

The ***Decision Tree*** model was used to identify non-linear relationships and interaction patterns between variables while maintaining high interpretability. It enables the visualization of decision rules that lead to a classification, making it useful for understanding how individual features contribute to the model's predictions.

The selected parameters directly influence the model's complexity and ability to generalize. The parameter **criterion** defines how the quality of each split is measured, while **max_depth** and **min_samples_split** control the structure of the tree, balancing bias and variance. Allowing the tree to expand fully provides insight into how easily the model can overfit when not constrained.

Adjusted parameters:

- **criterion = gini** — evaluates the quality of splits based on class impurity, guiding how branches are divided.

- **max_depth = None** — allows unrestricted tree growth, meaning the tree continues splitting until leaves are pure or contain fewer samples than defined by min_samples_split.

- **min_samples_split = 2** — sets the minimum number of samples required to divide a node, influencing how detailed the splits become.

K-Nearest Neighbors (KNN)

The ***K-Nearest Neighbours*** algorithm was applied to evaluate performance in a ***distance-based classification approach***. It classifies new samples based on the majority class of their nearest neighbours in the training data.

This method is intuitive and effective when the feature space is well-scaled, but its performance depends heavily on the number of neighbours and the distance metric used.

Adjusted parameters:

- **n_neighbors = 5** — defines the number of nearest neighbours considered when classifying a sample.

- **metric = minkowski** — general distance metric; with **p = 2**, it corresponds to the Euclidean distance.

- **p = 2** — specifies the Euclidean distance function.

Naïve Bayes

The ***Naïve Bayes*** classifier was included to test a ***probabilistic model*** based on Bayes' theorem with the assumption of feature independence. Despite this simplification, the model often performs well on high-dimensional data and imbalanced datasets.

It estimates class probabilities directly from the training data, providing fast computation and a strong theoretical foundation for binary classification problems.

Adjusted parameters:

- **Model = GaussianNB** — assumes a normal distribution for continuous features, suitable for numeric input variables.

XGBoost

The ***XGBoost*** algorithm is a ***gradient boosting ensemble method*** that builds sequential trees, each correcting the errors of the previous one. It is designed for performance and scalability, offering regularization techniques that prevent overfitting.

The model was fine-tuned to improve the learning rate and generalization capability, showing strong performance in capturing complex, non-linear relationships.

Adjusted parameters:

- **n_estimators = 500** — number of boosting rounds (trees).
- **learning_rate = 0.05** — controls the contribution of each tree to prevent overfitting.
- **max_depth = 6** — limits tree depth to balance complexity and generalization.
- **subsample = 0.8** and **colsample_bytree = 0.8** — introduce randomness to reduce overfitting.
- **reg_lambda = 1.0** — L2 regularization to improve model stability.

LightGBM

The ***LightGBM*** algorithm is a ***gradient boosting framework*** that uses histogram-based learning for high-speed training and efficient memory usage. It handles large datasets effectively while maintaining high predictive performance.

The model was configured to ensure balanced learning, with a moderate learning rate and tree complexity to optimize precision and recall.

Adjusted parameters:

- **n_estimators = 700** — number of trees used in the ensemble.
- **learning_rate = 0.05** — controls the contribution of each tree to learning.
- **num_leaves = 31** — defines the maximum number of leaves per tree, controlling complexity.
- **subsample = 0.8** and **colsample_bytree = 0.8** — introduce randomness to enhance generalization.
- **reg_lambda = 1.0** — applies L2 regularization to prevent overfitting.

MLP (Neural Network)

The ***Multi-Layer Perceptron (MLP)*** is a ***feed-forward artificial neural network*** designed to capture complex, non-linear relationships in the data. It consists of fully connected layers that transform inputs through activation functions to model higher-level abstractions.

The model was configured with two hidden layers and trained using the Adam optimizer, enabling adaptive learning and efficient convergence. Early stopping was implemented to prevent overfitting by halting training when validation performance stopped improving.

Adjusted parameters:

- **hidden_layer_sizes = (128, 64)** — defines two hidden layers with 128 and 64 neurons respectively.
- **activation = relu** — non-linear activation function improving learning capacity.
- **solver = adam** — adaptive optimizer suitable for large datasets.
- **learning_rate_init = 0.001** — controls the initial learning speed.
- **alpha = 0.0001** — L2 regularization term preventing overfitting.
- **batch_size = 256** — determines the number of samples per gradient update.
- **max_iter = 100** — limits the number of training epochs.
- **early_stopping = True** — stops training when validation score ceases to improve.

Evaluation & Interpretation

We analyzed multiple metrics:

- Accuracy, Precision, Recall, F1-score, and ROC-AUC.
- Confusion matrix to interpret false positives/negatives.
- ROC curve and AUC to compare discriminative ability.

Main Results

- ***Random Forest*** and ***XGBoost*** achieved the best overall results (*F1-score* and *AUC* ≥ 0.90).
- ***Logistic Regression*** maintained a good balance between accuracy and interpretability, with lower recall but easily interpretable coefficients.
- ***Naïve Bayes*** showed the weakest performance among all models.
- The most influential variables included: duration, previous, poutcome, contact, month, and socioeconomic features (age, job, education).

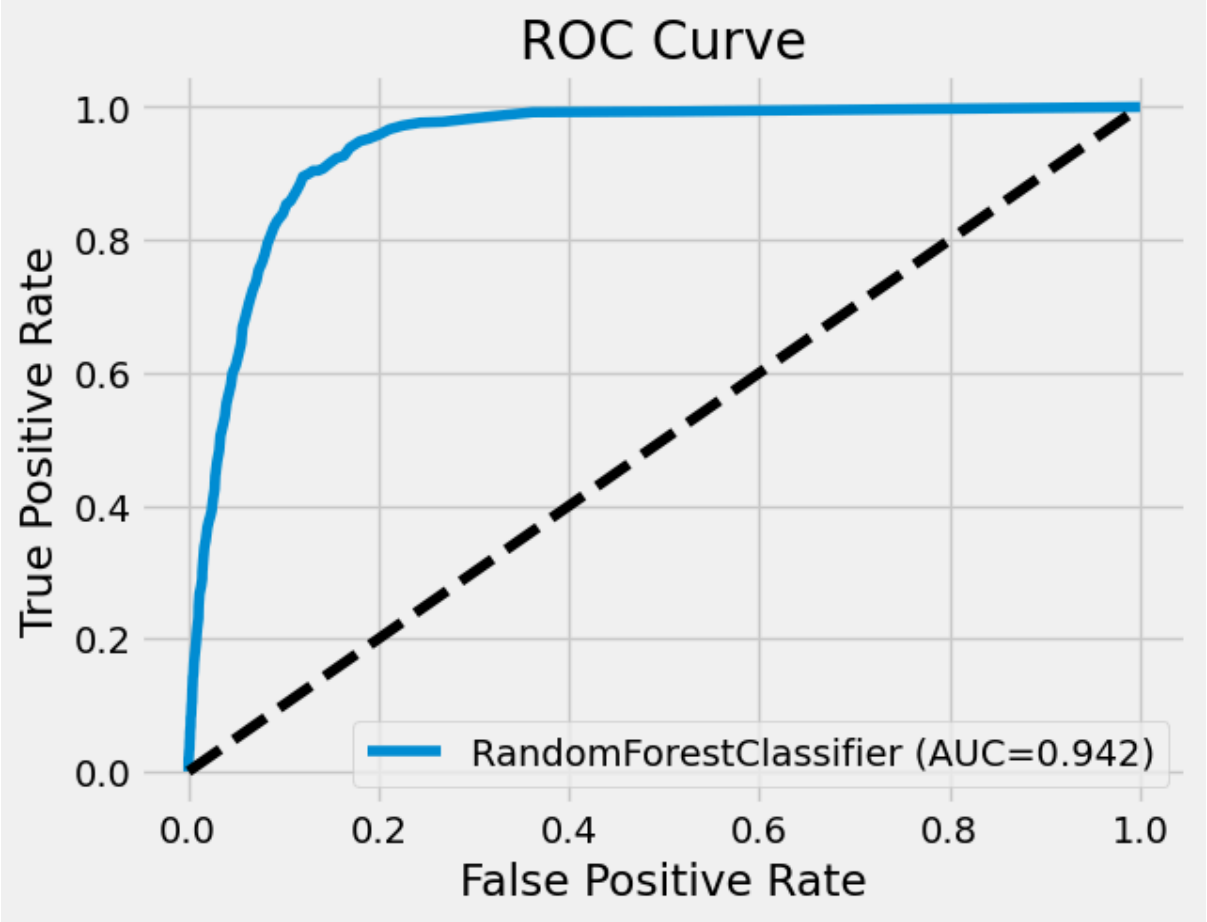
---

1. Random Forest

Random Forest presented the best overall balance between recall, precision, and F1-score.

- High recall shows that the model successfully identifies most customers who actually subscribe, reducing false negatives.
- Precision remained satisfactory, meaning most predicted positives are genuinely interested clients.
- A high ROC-AUC demonstrates excellent discriminatory power between clients who accept or reject the offer.
  Interpretation: Robust and generalizable, this model is ideal for practical deployment, maximizing correctly identified customers without introducing excessive noise. It provides the best trade-off between sensitivity (*recall*) and reliability (*precision*).

Threshold = 0.3	Precision	Recall	F1-Score	Support
0	0.97	0.92	0.94	7062
1	0.55	0.79	0.65	897



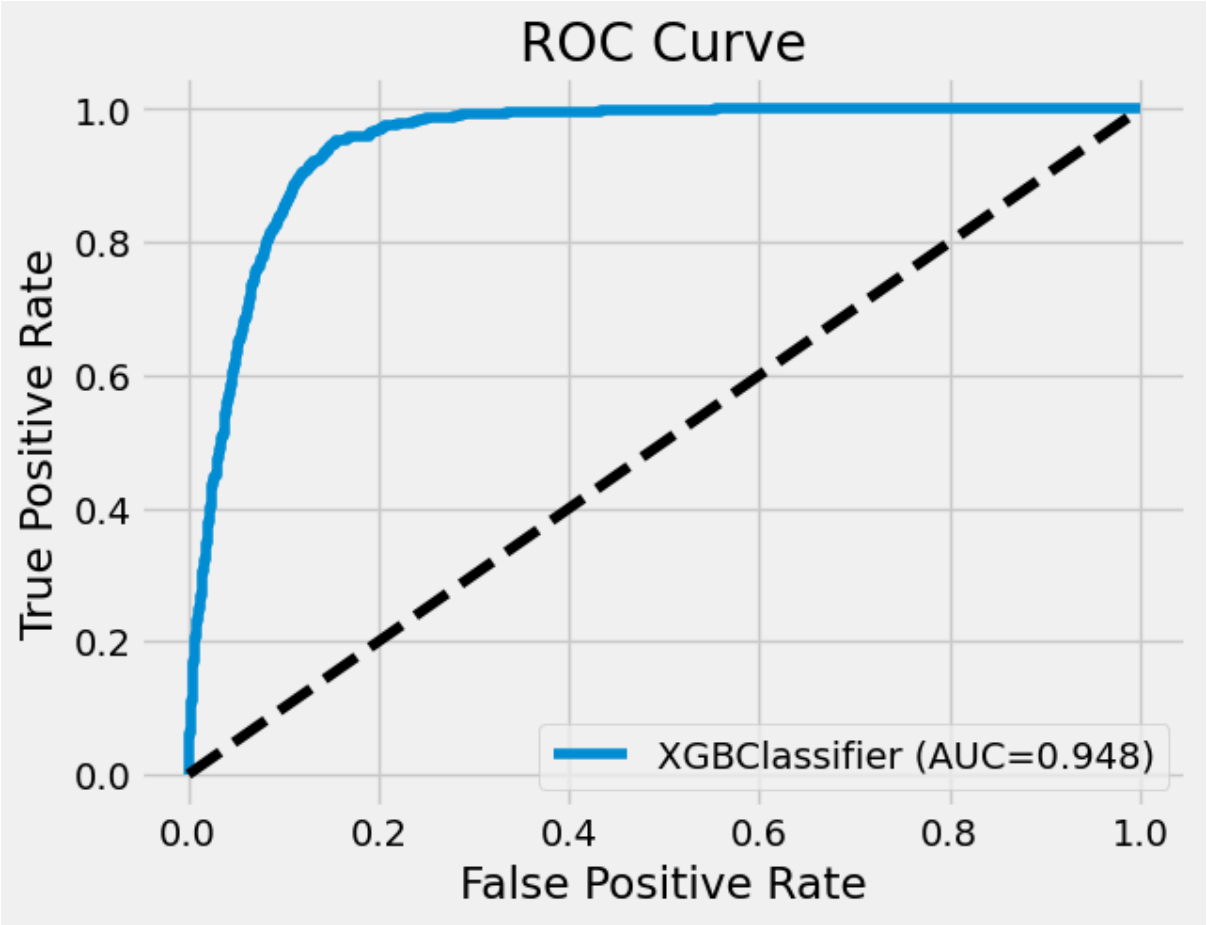
---

2. XGBoost

XGBoost achieved results very close or slightly superior in AUC, reflecting its ability to capture complex, non-linear patterns in customer behavior.

- It showed high recall, similar to Random Forest, with slightly higher precision in some runs.
- The boosting process iteratively corrects previous errors, giving it greater predictive power.
- However, it is computationally heavier and more sensitive to hyperparameter settings.
  Interpretation: XGBoost is the most predictively powerful model, delivering the best raw performance though requiring more tuning and offering lower interpretability. It is ideal when the goal is to maximize final predictive performance, even at higher complexity.

Threshold = 0.3	Precision	Recall	F1-Score	Support
0	0.97	0.93	0.95	7062
1	0.57	0.76	0.65	897



Conclusions & Recommendations

- The data mining process followed the CRISP-DM methodology effectively.
- The data preparation and modeling pipeline is robust, consistent, and reproducible.
- The best models (Random Forest and XGBoost) achieved an excellent balance between precision and recall, suggesting real-world applicability.
- A higher recall model is preferable, as it demonstrates greater ability to capture genuinely interested customers (more true positives) and thus superior performance in identifying sales opportunities.
- The number of calls made is less significant; therefore, even if the model produces some false positives, the accuracy rate remains acceptable.
- Recognized limitations: initial data imbalance and possible ***data leakage*** due to the inclusion of *duration* in benchmark tests.