PROJECT 2 IA

Pedro Borges – up202207552

Lucas Faria – up202207540

Alexandre Lopes - up202207015

Project Specification

- This project applies **supervised machine learning** to a **binary classification** problem: detecting **fraudulent transactions** in e-commerce data.
- The goal is to build models that can accurately classify transactions as **fraudulent (1)** or **legitimate (0)** based on labeled features.
- There were provided **two datasets** with a total of more than **1.5M rows**. These included several information's about the transactions such as the transactions amount, user age, shipping address, etc.

Related Work

- “Fraud Detection Dataset” on Kaggle
  This dataset provides labeled e-commerce transaction data for binary classification of fraudulent (1) vs. legitimate (0) transactions. It serves as the foundation for our project.

  https://www.kaggle.com/datasets/kevinvagan/fraud-detection-dataset
- GitHub Repository Linked to the Dataset
  The associated GitHub repository offers code examples for data preprocessing, model training, and evaluation, which have informed our methodology.
- https://github.com/slothislazy/Fraud-Detection-in-E-Commerce

- "Instance-Dependent Cost-Sensitive Learning for Detecting Transfer Fraud”
  This study introduces cost-sensitive classification models tailored for fraud detection, emphasizing the importance of considering misclassification costs in model training.
- https://arxiv.org/abs/2005.02488

- "Instance-Dependent Cost-Sensitive Learning for Detecting Transfer Fraud“ Detection of Fraudulent Sellers in Online Marketplaces using Support Vector Machine Approach”
  This research focuses on identifying fraudulent sellers in e-commerce platforms using Support Vector Machines, demonstrating the applicability of SVMs in fraud detection scenarios.”
  This study introduces cost-sensitive classification models tailored for fraud detection, emphasizing the importance of considering misclassification costs in model training.
- https://arxiv.org/abs/1805.00464

Tools Used and Pipeline Followed

- This project is developed in **Python** using **Pandas** for data manipulation, **Numpy** for numerical operations, **Scikit-learn** for implementing and evaluating models, **Catboost** and **XGBoost** as additional classifiers for our models, **Matplotlib** and **Seaborn** for data visualization.
- With the help of these tools, we followed the following **pipeline**:
    - Addressing the problem and identifying the target variables;
    - Merging the datasets;
    - Select random sample of the data to use in the models (20%);
    - Data preprocessing:
        - Removing non-analytical columns;
        - Deal with NA/NULL values;
        - Find outliers, strange values and data analysis to detect useful patterns;
    - Implement the algorithm using a training, validation, test approach;
    - Improve the results using several methods such as feature selection, GridSearchCV and dataset balancing;
    - Comparing the results.

Data Preprocessing and Data Analysis

- We **removed** the Transaction ID, Customer ID and IP Address. These columns had values very dispersed, with no analytical value, that didn't contribute to achieve our goal.
- The **NA values** were filled using a **probability-based imputation**. This approach leverages the distribution of known values within the dataset to generate plausible replacements for missing entries.
- We identified some **outliers** in the Transaction Amount that were removed.
- We also identified some **strange** ages. We eliminated the ones below 0 because they are impossible. We maintained some suspicious ages (below 18) because we didn't have information about the platform age restrictions.
- After the analysis we concluded that the Account Age Days, Transaction Hour and Address Match were the **best features**.

Algorithms

- The ML algorithm used in the Pipeline were:
  - **K-Nearest Neighbors:**
    - Good baseline algorithm.
    - It makes predictions based on the **similarity** between data points. It **does not need a training phase**; just stores data and calculates as needed.
    - We used **5 neighbors** as the default parameter to avoid dilution of the impact of fraudulent cases when the number of neighbors is high.
  - **Decision Tree:**
    - This algorithm is **a rule-based model** that is easy to interpret and visualize and can handle both numerical and categorical data.
    - It is also a good base line to compare with other algorithms.
    - We only used the balanced option in the **class weight** parameter to ensure the weight of fraudulent and non-fraudulent transactions were the same.
  - **Support Vector Machine:**
    - Focuses on finding the **optimal boundary (hyperplane)** that best separates fraudulent and legitimate transactions.
    - This algorithm often provides better results.
    - Used the linear SVM for the main dataset (normal SVM doesn’t do well with big datasets) and then used both for the balanced dataset.
    - We only increased the **maximum number of iterations** of the algorithm to enhance the result and used the balanced option in the class weight parameter.
  - **Logistic Regression:**
    - **Linear model** that estimates the probability of a transaction being fraudulent or legitimate using a **logistic function**.
    - It is simple but good to compare with more complex algorithms.
    - Similarly to the SVM algorithm, we only used the balanced option in the **class weight** parameter to ensure the weight of fraudulent and non-fraudulent transactions were the same.

Algorithms - 2

- **CatBoost Classifier:**
    - Is a **gradient boosting algorithm** that performs especially well on **tabular datasets** like the one used in this fraud detection task.
    - It is known for **its strong performance with minimal preprocessing** and is **robust to class imbalance** when using class weights.
    - We used the following **parameters** as default: iterations=500, learning_rate=0.1, depth=6, random_seed=1, verbose=0, class_weights=[1, 12]
    - These parameter help to improve the dataset balance and give a good default value before doing more improvements.
- **XGBoost Classifier:**
    - Highly efficient implementation of **gradient boosting**, optimized for **speed and accuracy**.
    - We used the following **parameters**: n_estimators=500, learning_rate=0.1, max_depth=6, random_state=1, scale_pos_weight=scale_pos_weight, use_label_encoder=False, eval_metric='logloss', n_jobs=-1.
    - It handles class imbalance through the **scale_pos_weight** parameter, which is useful for fraud detection.
    - The **n_jobs** is used to improve the execution speed of the algorithm allowing the use of multithreading.
- **Random Forest Classifier:**
    - **Ensemble learning method** that constructs **multiple decision trees** and outputs the mode of their predictions.
    - It is effective for **tabular data** and can handle class imbalance using the **class_weight=balanced** parameter.
    - We used the default values for most of the parameters except the **class weight** (balanced) and **n_jobs** (-1).

Evaluation of the models

- First try before improvements:
  - Overall, the results were **suboptimal**, with models favoring **non-fraudulent transactions** due to dataset imbalance.
  - While accuracy was high, the detection of fraudulent cases remained **low**.
  - **Best Accuracy: 95%** (K-Nearest Neighbors, Random Forest Classifier, Logistic Regression, Support Vector Machine).
  - **Best Precision: 100%** (Support Vector Machine – without the balanced option).
  - **Best Recall: 65%** (Logistic Regression, Support vector Machine – both with the balanced option) .
  - **Best F1 Score: 35%** (CatBoost).
  - Without the improvements, the **precision and recall values were very different with a big discrepancy.** These very different values reflect in the **low F1 Scores**

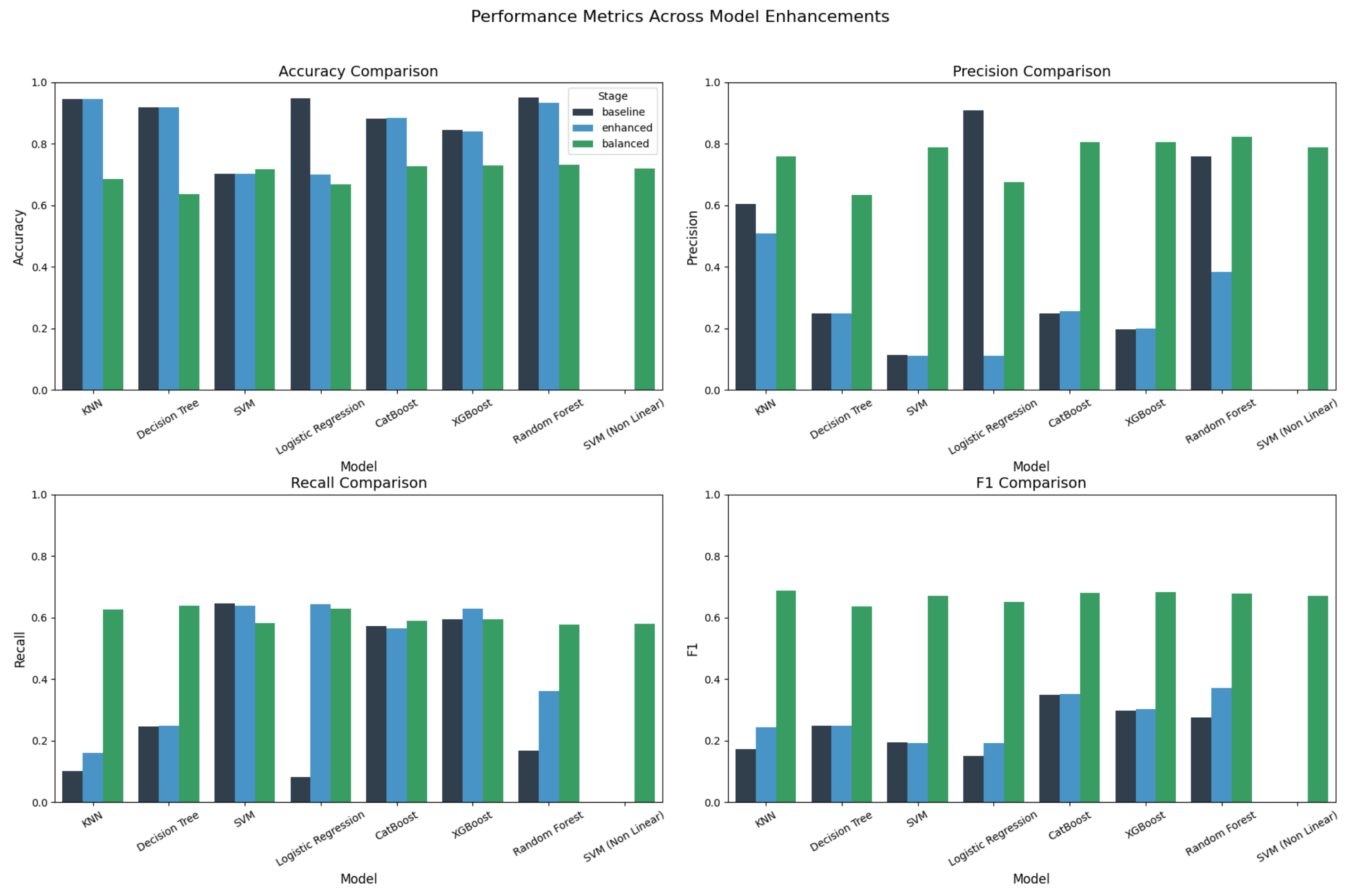
Evaluation of the models - 2

- Second try after selecting the best features and using GridSearchCV:
  - These changes had **minimal impact** on the results, showing little to no improvement.
  - **Best Accuracy: 95 %** (K-Nearest Neighbors)
  - **Best Precision: 80%** (K-Nearest Neighbors – after GridSearchCV with focus on accuracy)
  - **Best Recall: 65%** (Support Vector Machine, Logistic Regression)
  - **Best F1-Score: 37%** (Random Forest)
  - Despite these improvements, **recall and precision remained highly imbalanced**, leading to **significant discrepancies** between the two. This imbalance is reflected in the **low F1-scores**, highlighting the need for further optimization.

Evaluation of the models - 3

- Third try after balancing the dataset:
  - We balanced the dataset removing the excessive number of non-fraudulent transactions.
  - The results shown a big improvements compared to the other tries.
  - **Best Accuracy: 73%** (Catboost, XGBoost, Random Forest).
  - **Best Precision: 83%** (Random Forest).
  - **Best Recall: 64%** (Support Vector Machine - Linear, Logistic Regression, Decision Tree).
  - **Best F1-Score: 69%** (Random Forest, XGBoost, CatBoost, K-Nearest Neighbors).
  - At first glance, the results may seem worse when focusing solely on the highest accuracy, precision, and recall values. However, the gap between precision and recall across different algorithms has significantly narrowed, leading to **much improved F1-scores**, which better reflect overall model performance.

Conclusions



- After executing the pipeline, we can conclude that:
  - **Balancing** the dataset proved to be a **crucial step** in improving model performance. Only after addressing this issue did our models begin to deliver **meaningful results**.
  - Interestingly, some of the simplest models, such as **K-Nearest Neighbors** and **Random Forest**, produced **surprisingly strong results**, demonstrating their potential usefulness despite their simplicity.
  - While **XGBoost** and **CatBoost** consistently ranked among the top-performing algorithms both before and after improvements, the performance gap between them and the simpler models was **not substantial enough** to justify their higher computational cost and complexity.