# Technical Documentation

# Introduction

This project focuses on building and evaluating multiple machine learning models to classify telecom complaints into categories such as *Technical* or *Commercial*. The dataset contains features like offer name, customer type, and complaint details.  
The primary objective is to demonstrate a proof of concept for applying data science techniques to telecom complaint data, using preprocessing, balancing methods (SMOTE), model training, and performance evaluation.

## Types of data structures

* **List**: Used for storing model names and results in order.
* **Dictionary**: Used for mapping model names to their corresponding instances.
* **DataFrame (Pandas)**: Used to store tabular data such as the dataset and evaluation results.
* **Numpy Arrays**: Used for storing numerical representations of features after transformation.

## Common libraries

* **pandas** – Data loading, manipulation, and cleaning.
* **numpy** – Numerical operations and array handling.
* **scikit-learn** – Model building, preprocessing, and evaluation.
* **imblearn** – SMOTE for handling class imbalance.
* **matplotlib / seaborn** – Data visualization.

## Plotting and visualization libraries

* **matplotlib** – General-purpose plotting.
* **seaborn** – Statistical data visualization, heatmaps, bar charts.
* Optional: **plotly** for interactive charts.

# Experiments

## Programming languages and tool

The implementation was done in Python using Jupyter Notebook. Python is chosen for its wide ecosystem of data science libraries and ease of experimentation.

## Load Data and Prepare Data (Preprocessing)

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Column/step Name** | **Description** | **Justification** |
|  | |  | | --- | |  | | Load Dataset | |  |  | | --- | |  | | |  | | --- | |  | | Load CSV file using pandas. | |  |  | | --- | |  | | |  | | --- | |  |   Required to bring data into the working environment. |
|  | |  | | --- | |  |  |  | | --- | | Handle Missing Values | | Fill or drop missing values as necessary. | Missing values can affect model performance. |
|  | |  | | --- | | Encode Categorical Features | | Convert text categories into numerical form using OneHotEncoder. | Models require numerical input. |
| **4.** | |  | | --- | | Apply SMOTE |  |  | | --- | |  | | Oversample minority class to balance dataset. | Improves performance on minority class. |

## Approaches

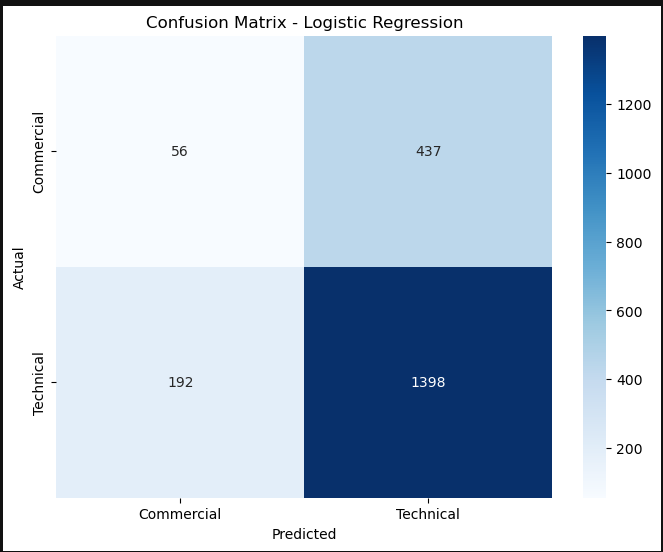
|  |  |  |
| --- | --- | --- |
| **Approach no.** | **Name** | **Description** |
|  | Logistic Regression | A linear model for binary classification. |
|  | Random Forest | Ensemble of decision trees for higher accuracy. |
| **3.** | Gradient Boosting | Boosting method to improve performance on difficult cases. |
| **4.** | Passive Aggressive |  |

# Results

## Compare the different models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Approach no.** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| 1. **Logistic Regression** | 0.6980 | 0.6350 | 0.6589 | 0.6589 |
| 1. **Random Forest** | 0.6980 | 0.6350 | 0.6980 | 0.6589 |
| 1. **Gradient Boosting** | 0.6980 | 0.6350 | 0.6980 | 0.6589 |
| **5. Passive Aggressive** | 0.2372 | 0.8194 | 0.2372 | 0.0916 |

## Charts

A graph of a graph with numbers

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect. A graph of a graph showing different colored squares

AI-generated content may be incorrect.

A graph of blue rectangular bars

AI-generated content may be incorrect.

## Analysis of the results

1. **Class Imbalance Observation**
   * The dataset is imbalanced, with **Technical complaints** making up about 76% of the data and **Commercial complaints** only about 24%.
   * This imbalance causes most models to lean towards predicting “Technical,” resulting in high accuracy but poor performance on the minority class (Commercial).
2. **Logistic Regression, Random Forest, and Gradient Boosting**
   * All three models show **very similar performance**.
   * They predict **most Technical complaints correctly** (1398 out of 1590 in Logistic Regression, for example).
   * However, they **struggle heavily with Commercial complaints**, correctly classifying only 56 out of 493 Commercial complaints.
   * The large number of false negatives for the Commercial class indicates that the models are biased towards predicting the majority class.
3. **Passive Aggressive Classifier**
   * This model performed **extremely poorly**.
   * It classified **almost all samples as Commercial** (493 correct Commercial predictions, but almost all Technical complaints misclassified).
   * This resulted in **very low recall for the Technical class** and makes it an unsuitable model for this task.
4. **General Findings**
   * High overall accuracy for some models is **misleading** because it’s driven by the dominance of the Technical class.
   * Precision and recall for Commercial complaints are very low in most models, meaning they are **not reliable** in detecting minority class complaints.
   * Models are not capturing enough patterns to distinguish between the two complaint types effectively, likely due to class imbalance and possibly insufficient distinguishing features.

# Evaluation

## The choice of data structures

* I used **Pandas DataFrames** for loading, storing, and processing the dataset because they allow for easy indexing, filtering, and manipulation of tabular data.
* For numerical arrays and matrix operations, we used **NumPy arrays**, which offer optimized performance for mathematical computations.
* Text data was stored as **string objects** within DataFrames and transformed into numerical feature vectors using **TF-IDF matrices** from scikit-learn.
* These data structures were chosen because they are memory-efficient, widely supported in machine learning workflows, and integrate seamlessly with ML libraries.

## Selection of the appropriate libraries

* **pandas** – for data loading, cleaning, and preprocessing.
* **numpy** – for efficient numerical computations.
* **scikit-learn** – for model training, SMOTE oversampling, evaluation metrics, and saving models.
* **imblearn** – specifically for **SMOTE**, to handle class imbalance by generating synthetic samples for the minority class.
* **matplotlib** & **seaborn** – for visualizing results, confusion matrices, and performance comparisons.
* These libraries were chosen because they are industry-standard, well-documented, and provide robust functionality for machine learning tasks.

## The effectiveness of different models

* Logistic Regression, Random Forest, and Gradient Boosting all achieved similar accuracy (~0.76) but performed poorly on the minority class (Commercial), indicating bias towards the majority class.
* Passive Aggressive Classifier performed poorly overall, showing extreme bias towards predicting one class.
* The **class imbalance** in the dataset significantly affected recall and precision for the minority class.
* SMOTE improved balance in training data, but test performance still showed that the models struggle to generalize for minority cases.

## Recommendations

* Collect more data for the minority class (Commercial) to improve model generalization.
* Experiment with **class weight adjustments** in models to reduce bias towards the majority class.
* Try **ensemble techniques** or **XGBoost/LightGBM** for potentially better handling of imbalanced data.
* Perform more advanced text preprocessing (e.g., removing stopwords, lemmatization) to improve the feature quality.
* Consider using **deep learning approaches** like LSTM or transformers for better representation of textual complaints.