**Comparing Classifiers: Bank Marketing Campaign Analysis**

**Project Overview**

This project applies various classification methods to a business problem: determining whether clients will subscribe to a term deposit after being contacted through phone-based marketing campaigns. The dataset is derived from the UCI Machine Learning Repository and contains data on direct marketing campaigns conducted by a Portuguese banking institution.

The project implements and compares multiple machine learning algorithms using various metrics to assess model performance.

**Objective**

The primary objective is to compare the results of different classification algorithms and determine which is the best for predicting client subscriptions. This prediction will allow the bank to focus resources on clients most likely to subscribe, optimizing marketing efforts.

**Key Business Objective**

* **Goal**: Apply and compare various classifiers (K-Nearest Neighbors, Logistic Regression, Decision Trees, and SVM) to improve marketing efficiency by predicting client subscription likelihood.
* **Impact**: Increase conversion rates and reduce costs by focusing marketing efforts on high-potential clients.

**Datasets**

This project uses four datasets from the UCI Machine Learning Repository:

1. **bank-additional-full.csv** – Full dataset with 41,188 examples and 21 features.
2. **bank-additional.csv** – 10% sample of the full dataset, containing 4,119 examples and 21 features.
3. **bank-full.csv** – Older version of the dataset with 45,211 examples and 17 features.
4. **bank.csv** – 10% sample of the older version, containing 4,521 examples and 17 features.

The datasets are available [here](https://archive.ics.uci.edu/dataset/222/bank+marketing). **Please download the datasets manually** and place them in the appropriate directory before running the project.

**Target Variable:**

* y – Whether the client subscribed to a term deposit (yes/no).

**Models Implemented**

Four machine learning models were implemented and evaluated on each dataset:

* **K-Nearest Neighbors (KNN)**
* **Logistic Regression**
* **Decision Tree**
* **Support Vector Machine (SVM)**

**Evaluation Metrics:**

The models were evaluated using:

* Accuracy
* Precision
* Recall
* F1 Score
* Support

**Project Structure**

The project is organized into the following steps:

1. **Exploratory Data Analysis (EDA)**:
   * Provides descriptive statistics and visualizations for understanding the distribution of key features and the target variable.
   * Visualizations include histograms, box plots, and count plots for class imbalance.
2. **Data Preprocessing**:
   * Handling categorical features using one-hot encoding.
   * Splitting the dataset into training and testing sets.
   * Scaling numerical features for models like KNN and SVM.
3. **Model Implementation**:
   * Implementation of KNN, Logistic Regression, Decision Tree, and SVM.
   * Models are trained on the training set and evaluated on the test set.
4. **Model Evaluation and Comparison**:
   * Model performance is compared across datasets using grouped bar plots for accuracy, precision, recall, F1-score, and support.
   * Visualizations are provided to give a clear comparison of each model’s performance on the various metrics.

**Visualizations**

The following visualizations are included to make the analysis more interpretable:

* **Descriptive Statistics**:
  + Distribution plots for key features (age, duration, campaign contacts).
  + Class imbalance visualization for the target variable y.
* **Inferential Statistics**:
  + Grouped bar plots for comparing accuracy, precision, recall, F1-score, and support across the four models.
  + Heatmap for visualizing the model accuracy comparisons.

**Results**

Across all datasets, **Logistic Regression** and **SVM** emerged as the most effective models. Both models performed well across different metrics (accuracy, precision, recall), and they were particularly good at handling the class imbalance between clients who subscribed and those who did not.

The key findings are:

* **Call Duration** was the most significant predictor of whether a client would subscribe. The longer the call, the more likely the client was to engage.
* **Logistic Regression** had the highest accuracy across most datasets and is recommended for deployment due to its simplicity and scalability.
* **SVM** also performed well but comes with higher computational costs, making it suitable for more complex datasets.
* **KNN** and **Decision Tree** showed good performance on smaller datasets but struggled with generalization on larger datasets.

**Next Steps**

* **Address Class Imbalance**: Techniques like SMOTE or adjusting class weights should be used to handle the imbalance in the target variable.
* **Improve Call Engagement**: Since call duration is a strong predictor, efforts should focus on increasing the length and quality of client interactions during calls.
* **Deploy Logistic Regression**: Given its high accuracy and computational efficiency, Logistic Regression is recommended for deployment in the bank’s marketing campaigns.

**Recommendations**

* The bank should focus on using **Logistic Regression** or **SVM** models for client subscription prediction due to their strong performance.
* Further work should address the **class imbalance** by applying techniques like oversampling or adjusting the model evaluation metrics to favor minority class detection.
* **Call duration** was identified as a key predictor for term deposit subscription, suggesting that efforts should focus on engaging clients in longer conversations during marketing calls.

**How to Run the Project**

1. **Clone the repository**:

bash

Copy code

git clone https://github.com/stanleykelman/Comparing-Classifiers.git

1. **Install dependencies**: Install the necessary Python libraries using requirements.txt (or manually if there’s no file):

bash

Copy code

pip install -r requirements.txt

1. **Run the Jupyter Notebook**: Open the project in Jupyter Notebook and run the cells sequentially to see the analysis and visualizations:

bash

Copy code

jupyter notebook

**Dependencies**

* Python 3.x
* Jupyter Notebook
* Pandas
* Scikit-learn
* Matplotlib
* Seaborn
* Numpy

**Contributing**

Feel free to submit pull requests or open issues if you have any suggestions for improvements or additional features.

**License**

UC Berkeley

**Contact**

If you have any questions or feedback, please contact me at:

* **GitHub**: [stanleykelman](https://github.com/stanleykelman)
* **Email**: stanleykelman@gmail.com

**Thank you for checking out my project! I look forward to your feedback and contributions.**