Report of mini project 1

Nazmus Sakib
nazmus.x.sakib@abo.fi

Problem statement:

The project aims to analyze a dataset related to direct marketing campaigns conducted by a banking institution via phone calls to clients. The primary objective is to develop predictive models using two different machine learning algorithms to forecast whether a client will respond positively or negatively to the campaign, i.e., whether they will subscribe ('yes') or not ('no') to a bank term deposit. The tasks include data preprocessing, selecting relevant features for the models, processing the data appropriately, training the models to achieve at least 80% accuracy, visualizing their performance, and comparing the results to understand any differences between the models. Here is the google colab link:
https://colab.research.google.com/drive/1-RMPlAzjGA3afuhl0DD3YfD06eQz1CZr?usp=sharing

Data processing

- **Reading the dataset**:
  - The code begins by reading the dataset from a CSV file. This dataset contains information about marketing campaigns conducted by a banking institution.
  - I read the dataset from a CSV file, specifying ';' as the delimiter. This delimiter indicates that the dataset is structured using semicolons to separate values within each row.
- **Handling missing values:**
  - Next, I checked for missing values in the dataset.Although I identified missing values, I didn't explicitly handle them in this snippet because there was no missing value found.
- **Encoding categorical variables:**
  - The dataset likely includes categorical variables such as job type and marital status that need to be converted into numerical format for machine learning algorithms to process.
  - To achieve this, I selected these string columns and applied encoding techniques to convert their values into numerical representations. This step is crucial because most machine learning algorithms require numerical input.
  - I used OrdinalEncoder to encode feature columns
- **Label encoding the target variable:**
  - Alongside encoding categorical features, I also performed label encoding on the target variable ('y'), which indicates whether a client subscribed to a bank term deposit ('yes' or 'no').
  - Label encoding assigned a unique numerical label to each category in the target variable, facilitating its use in machine learning models.

- **Correlation analysis:**
    - I calculated the correlation between different features in the dataset to understand the relationships between them. Correlation measures the strength and direction of the linear relationship between two variables.
    - Visualizing the correlation matrix as a heatmap allowed me to identify patterns and relationships between features, guiding feature selection and model building.
- **Feature selection based on correlation:**
    - After analyzing feature correlations, I selected features with correlation coefficients above a certain threshold (greater than 0.01 in this case) for further analysis.
    - Features with higher correlation coefficients were prioritized for inclusion in machine learning models as they were more likely to have predictive power.
    - Dropped the ‘duration ’ column because the duration is not known before a call is performed.

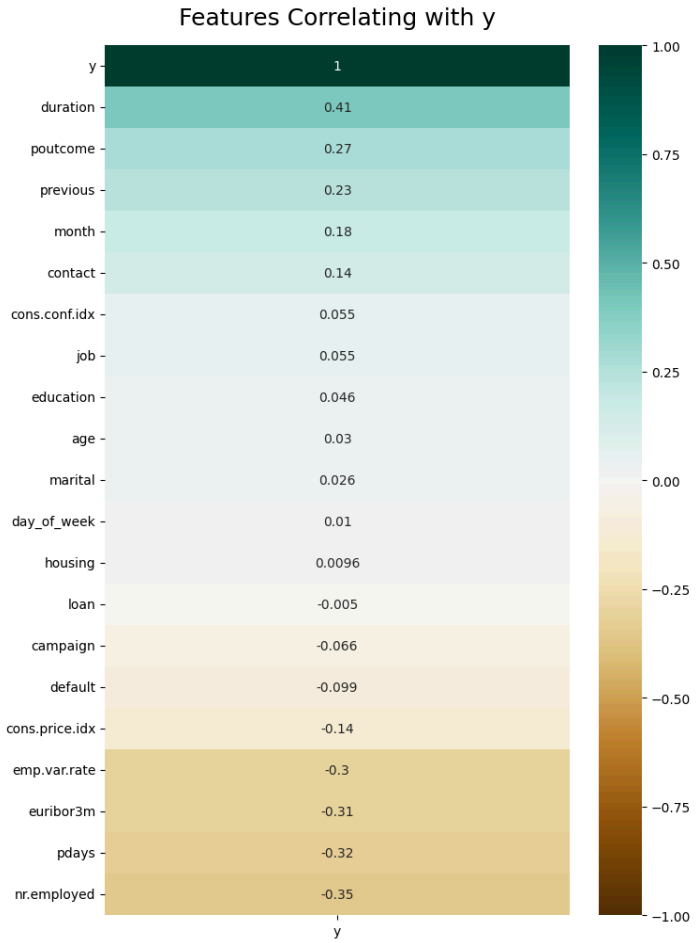


Fig: Sorted columns based on correlation coefficient (both positively and negatively correlated)

Modeling

Algorithms Selection:
I chose two different classification algorithms for this task: K Nearest Neighbors (KNN) and Gradient Boosting Classifier (GB).

Reasons for Selection:

- **K Nearest Neighbors Classifier**: It's a simple yet effective algorithm that can handle non-linear relationships in data. Given the nature of the problem and the potential non-linearity in the relationships between features and the target variable, KNN seemed like a suitable choice.
- **Gradient Boosting Classifier**: Gradient Boosting is an ensemble learning technique known for its high predictive power. It builds multiple weak learners sequentially, each focusing on the mistakes made by the previous learners. This makes it robust and capable of capturing complex patterns in the data.

Performing the Models:

- **Model Training**: I split the dataset into training and testing sets, and then trained both KNN and GB classifiers on the training data.
- **Model Evaluation**: After training, I evaluated the models' performance on the test data using various metrics such as precision, recall, F1-score, and accuracy.
- **Visualizing Performance:** To compare the performance of both models, I plotted ROC curves for each model using RocCurveDisplay from scikit-learn.

Improving Model Performance: I was not able to achieve the required accuracy at the first training round. I had to fix the class imbalance to achieve 80% accuracy.

- **Handling Class Imbalance by Data Sampling**: Recognizing the imbalance in the dataset, I addressed it by employing the Synthetic Minority Over-sampling Technique (SMOTE). This technique helped balance the distribution of the target variable by generating synthetic samples for the minority class ('yes' subscriptions). As a result, both the KNN and GB classifiers were trained on a more balanced dataset, leading to improved generalization and performance.
- **Feature Engineering:** I calculated the correlation coefficients between different features in the dataset. By selecting features with correlation coefficients above a certain threshold (greater than 0.01), I aimed to include only those features that exhibited stronger relationships with the target variable. This helped reduce **dimensionality** and focus on the most **relevant** attributes for predictive modeling.

Achieving Required Accuracy: Both models achieved high accuracy scores after the first training round. KNN achieved an accuracy of approximately 88%, while GB Classifier achieved an accuracy of around 91%.

- **KNN**: In the classification report of K nearest neighbor, for class 0 (not subscribed), the model achieved a precision of 0.89, indicating that 89% of instances predicted as not subscribed were actually not subscribed. The recall for class 0 is 0.88, meaning that 88% of actual not subscribed instances were correctly classified. The F1-score for class 0 is also 0.88, indicating a balanced performance between precision and recall. Similarly, for class 1 (subscribed), the model achieved a precision of 0.88, indicating that 88% of instances predicted as subscribed were actually subscribed. The recall for class 1 is 0.89, meaning that 89% of actual subscribed instances were correctly classified. The F1-score for class 1 is also 0.88. The overall accuracy of the model is 0.88, indicating the proportion of correct predictions over all instances. The confusion matrix shows that the model correctly classified 6452 instances of class 0 and 6444 instances of class 1, while misclassifying 894 instances of class 0 and 830 instances of class 1.
- **GB** : In this classification report for the Gradient Boosting model, for class 0 (not subscribed), the precision is 0.89, indicating that 89% of instances predicted as not subscribed were actually not subscribed, while the recall is 0.94, indicating that 94% of actual not subscribed instances were correctly classified. The F1-score for class 0 is 0.91, reflecting a balanced performance between precision and recall. For class 1 (subscribed), the precision is 0.93, indicating that 93% of instances predicted as subscribed were actually subscribed, while the recall is 0.88, indicating that 88% of actual subscribed instances were correctly classified. The F1-score for class 1 is also 0.91. The overall accuracy of the model is 0.91, with 91% of correct predictions over all instances. The confusion matrix shows that the model correctly classified 6895 instances of class 0 and 6387 instances of class 1, while misclassifying 451 instances of class 0 and 887 instances of class 1.

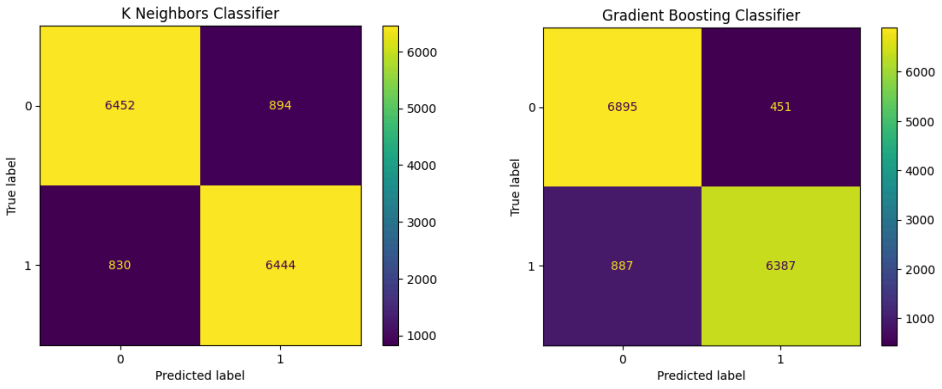
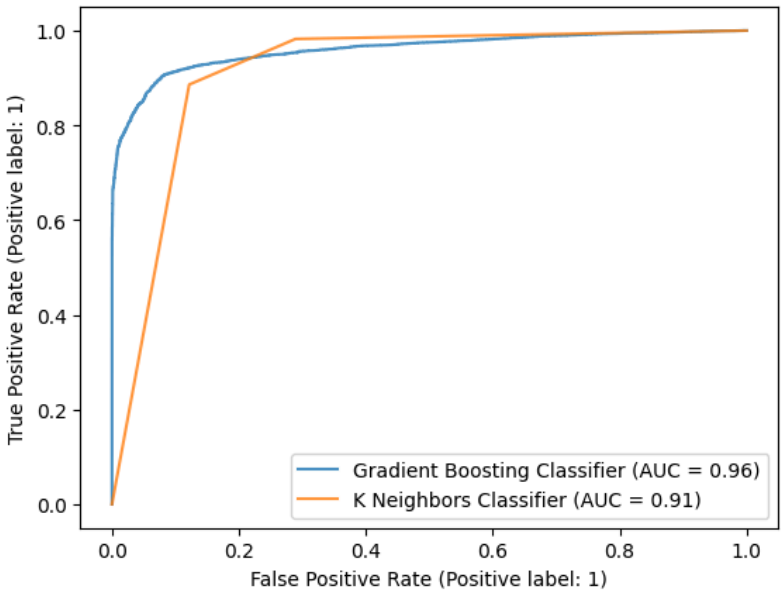


Fig: Confusion matrix of KNN and Gradient Boost classifier

- In analyzing the ROC curves, we can notice that the Receiver Operating Characteristic (ROC) curve for the Gradient Boosting Classifier (GB) exhibited a larger area under the curve (AUC) and was positioned closer to the y-axis in comparison to the ROC curve for the K Neighbors Classifier (KNN). This suggests that the GB classifier has better overall performance in distinguishing between the positive and negative classes. Additionally, the GB curve's proximity to the y-axis indicates higher true positive rates and lower false positive rates across various threshold values, emphasizing its superior discriminatory ability over the KNN classifier.

  In the ROC analysis, I observed that the area under the curve (AUC) for the Gradient Boosting Classifier (GB) is notably larger at 0.96, indicating its superior performance in distinguishing between positive and negative classes. Conversely, the AUC for the K Neighbors Classifier (KNN) is 0.91, reflecting slightly lower discriminatory power.



Conclusion:

The primary scientific bottleneck in this project was the challenge posed by the imbalanced nature of the dataset. The imbalance in the target variable ('yes' and 'no' subscriptions to term deposits) could lead to biased model predictions and reduced performance, especially for minority classes. To overcome this bottleneck, I employed the Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for the minority class, thereby balancing the class distribution and improving model performance.

In terms of algorithm performance, the Gradient Boosting Classifier (GB) outperformed the K Neighbors Classifier (KNN). The GB classifier exhibited higher precision, recall, and F1-score for both the positive ('yes') and negative ('no') classes, as well as a higher overall accuracy and

larger area under the ROC curve (AUC). This suggests that the GB classifier achieved better discrimination between the two classes and provided more accurate predictions overall. Therefore, in this scenario, the GB algorithm proved to be more effective for predicting term deposit subscriptions in the banking marketing campaign dataset.