Process details -

1. Feature Selection -
    a. Prints the shape and summary statistics of the loaded data.
    b. Checks if there are any null values and prints the count of each.
    c. Plots histograms and scatterplots for a subset of columns to get a visual understanding of the distribution of values and relationship with the target variable.
    d. Removes outliers from the data using interquartile range (IQR) method.
    e. Converts some columns from categorical values to numerical values using a mapping.
    f. Drops the columns whose all values are same like 'poutcome' column.
    g. Plots histograms for the remaining columns.

2. Data preparation and splitting -
    a. The code imports the train_test_split function from sklearn.model_selection to split the input data x and the target variable y into training and test datasets.
    b. The function train_test_split is used to divide the data into training and test sets in a ratio of 67% to 33% respectively. The random_state parameter is set to 423 for reproducibility.

3. Logistic Regression -
    a. The code imports the LogisticRegression class from sklearn.linear_model
    b. A Logistic Regression model is created and trained on the training data using the fit method. The hyperparameter C is set to 0.05 and the solver used is "liblinear".
    c. The model's performance is evaluated on the test data by making predictions using the predict method and comparing the predictions with the actual target values using various metrics such as accuracy, precision, recall, F1-score, and AUC-ROC score.

4. Random Forest -
    a. The code imports the RandomForestClassifier class from sklearn.ensemble.

   b. A Random Forest model is created and trained on the training data using the fit method.
   c. The model's performance is evaluated on the test data by making predictions using the predict method and comparing the predictions with the actual target values using various metrics such as accuracy, precision, recall, F1-score, and AUC-ROC score.

5. Gradient Boosting Classifier -
   a. The code imports the GradientBoostingClassifier class from sklearn.ensemble.
   b. A Gradient Boosting model is created and trained on the training data using the fit method.
   c. The model's performance is evaluated on the test data by making predictions using the predict method and comparing the predictions with the actual target values using various metrics such as accuracy, precision, recall, F1-score, and AUC-ROC score.

6. Hyperparameter tuning for Gradient Boosting -
   a. The code uses GridSearchCV from sklearn.model_selection to perform a grid search for the best hyperparameters for the Logistic Regression, Gradient Boosting, Random Forest model.
   b. The grid search is performed using the training data and the best hyperparameters are chosen based on the accuracy score.

7. Handling the imbalance data -
   a. To handle imbalanced data we have used Stratified K-Fold cross-validation class from the sklearn.model_selection module.
8. Result -
   a. In the learning curves we can see that LR and RF tries to overfit the data.
   b. These models have accuracy nearly 95%.
   c. AUC-ROC score of all the model is nearly 0.5-0.6 which is suggest that the model is unable to make a distinction between the two classes.

The limitations of the code could be:

1. The code is limited to binary classification, meaning it can only classify data into two categories. It can be challenging to expand this code to perform multi-class classification.
2. The code does not perform data preprocessing well which can result in suboptimal performance of the model due to the presence of outliers or skewed distributions in the data.
3. Other ML model can also be used to get better performance.

Possible future work could be:

1. Implement deep learning models.
2. Implement better feature selection and
3. Incorporate more advanced optimization algorithms, such as Adam or Adagrad, to improve convergence and training speed.
4. Experiment with different activation functions and architectures to see how they affect the performance of the model.