Introduction to Machine Learning (SS 2023)
Programming Project

Author 1
Last name:
First name:
Matrikel Nr.:

Author 2
Last name: Untermann
First name: Sven
Matrikel Nr.: 11906555

I. Introduction

The ability to identify fraudulent transactions is of great interest to the payments industry. In this project, we will utilize a binary classifier trained on a transactions dataset to detect fraud. The dataset consists of various features related to the transactions, along with the amount of each transaction. Our objective is to determine, based on these features, whether a transaction is fraudulent or not. The dataset contains 227,845 instances of transactions with 29 different features. Throughout this project, we will train and evaluate the classifier using these data

II. Implementation / ML Process

- The given dataset consists of transaction data with a binary classification task of identifying fraud transactions. Before selecting a model, it is important to analyze the data and address any issues, such as class imbalance.
- The dataset is highly imbalanced, with a majority of legitimate transactions (Class 0) and a minority of fraud transactions (Class 1). To balance the dataset, the "Class 1" samples can be oversampled using the resample function from scikit-learn. This creates a new dataset with an equal number of samples for both classes. The balanced dataset is then used for training the models.
- After balancing the dataset, we can proceed with selecting a suitable model. The script provides options for different algorithms, including Logistic Regression, Decision Trees, Random Forests, Multilayer Perceptron (Neural Network), and AdaBoost. Each algorithm is defined with its corresponding hyperparameter search space.
- The chosen algorithm family for this problem is ensemble methods, which includes Decision Trees, Random Forests, and AdaBoost. Ensemble methods combine multiple weak models to create a stronger model with improved generalization and robustness. These algorithms are suitable for this problem because they can handle complex relationships in the data and have the ability to learn from imbalanced datasets.
- The hyperparameters for each algorithm are chosen using grid search with cross-validation.

  This approach systematically tests different combinations of hyperparameters and selects the best performing set. The evaluation metric used for hyperparameter selection is ROC AUC (Receiver Operating Characteristic Area Under the Curve), which is a common metric for imbalanced classification problems.
- For the final model, the best hyperparameters obtained from grid search are used. The hyperparameters are as follows:

  Logistic Regression:
  C = 0.1,
  class_weight = {0: 0.1, 1: 0.9}
  Decision Tree:
  max_depth = 15,
  class_weight = 'balanced'
  Random Forest:
  n_estimators = 100,
  max_features = 'sqrt',
  class_weight = 'balanced'
  Multilayer Perceptron:
  hidden_layer_sizes = (50, 50),
  activation = 'relu',
  alpha = 0.0001
  AdaBoost:
  n_estimators = 100,
  estimator =DecisionTreeClassifier(max_depth=4)
- These hyperparameters are chosen based on their performance in terms of ROC AUC and accuracy scores on the testing data. The final models are trained using these hyperparameters.
- It is important to note that the provided script does not cover the entire data preprocessing and feature engineering pipeline. It assumes that the data is already preprocessed and ready for model training. In a real-world scenario, additional steps such as data cleaning, feature scaling, and feature selection may be required before training the models.

III. Results

- Describe the performance of your model (in terms of the metrics for your dataset) on the training and validation sets with the help of plots or/and tables.
- You must provide at least two separate visualizations (plot or tables) of different things, i.e. don't use a table and a bar plot of the same metrics. At least three visualizations are required for the 3 person team.

IV. Discussion

- Analyze the results presented in the report (comment on what contributed to the good or bad results). If your method does not work well, try to analyze why this is the case.
- Describe very briefly what you tried but did not keep for your final implementation (e.g. things you tried but that did not work, discarded ideas, etc.).
- How could you try to improve your results? What else would you want to try?

V. Conclusion

- Finally, describe the test-set performance you achieved. Do not optimize your method based on the test set performance!
- Write a 5-10 line paragraph describing the main take-away of your project.