Report on Spam Email Classification Using XGBoost Algorithm

Vikash Singh Darshan Jadhav

1 Algorithm Selection

For this case study, we chose to use an XGBoost classifier to predict whether an email is spam or not based on the provided features. XGBoost is an implementation of gradient boosted decision trees, which is an ensemble method that combines multiple weak learners (decision trees) into a strong learner.

A Python library "lazypredict" was employed to evaluate multiple machine learning models for the classification task. Through thorough analysis, we identified Random Forest with AdaBoost as the best fit. Hence, we implemented Gradient Boosting, a technique utilizing gradients to enhance learning rules, to optimize our classification model

Some key advantages of XGBoost that make it well-suited for this problem:

- Handles missing values in the data seamlessly when building the trees. This avoids the need for separate imputation preprocessing.
- Builds the model in a stage-wise fashion by adding trees sequentially. This helps prevent overfitting compared to a single large decision tree.
- Allows regularization parameters like `max_depth`, `min_child_weight`, `gamma`, etc. to control model complexity. We tuned these via grid search CV.
- Provides good out-of-the-box performance so minimal manual tuning is required beyond the regularization parameters.
- Computes second-order gradients to speed up the optimization process compared to traditional gradient boosting.

The main parameters we tuned using grid search cross-validation were:

- `n_estimators`: Number of boosted trees to fit. More trees can improve accuracy but increase training time. We tested on many different values but found 200 to provide a good balance.
- `learning_rate`: It is the shrinkage parameter that controls the contribution of each tree. Lower values make the model more conservative and prevent overfitting. Here, 0.1 value worked well.

- `max_depth`: Maximum depth of each tree. It is used to prevent overfitting. Here, 4 provided the best validation AUC.

2 Data Preprocessing

The dataset used for this classification task underwent the following preprocessing steps:

- **Data Loading:** The dataset was loaded from 'spamTrain.csv' and shuffled to prevent biases in the data.
- **Feature Selection:** Features and labels were separated, with features representing email attributes and labels indicating spam or non-spam classification.
- **Data Splitting:** The dataset was split into training and testing sets using a 70-30 split ratio. 70% of the data was used for training, and 30% was reserved for testing.
- **Missing Value Handling:** we thoroughly explored various strategies for handling missing values, including explicit imputation techniques. We diligently tested methods such as mean imputation, median imputation, mode imputation, and even advanced techniques like k-nearest neighbors imputation and multiple imputation. However, we found that XGBoost's inherent capability to handle missing values proved to be remarkably efficient. Leveraging this built-in feature eliminated the necessity for explicit imputation techniques, streamlining our workflow and enhancing the overall efficiency of our analysis

Normalization was also not necessary as the features were proportions in the range [0, 1].

3 Model Evaluation and Results

We evaluated our model using 10-fold cross-validation on the training data, followed by holding out 30% of the data for a final test set.

The cross-validation AUC was 0.910, indicating the model generalizes fairly well across different splits of the training data.

On the held-out test set, the model achieved an AUC of 0.89 and a true positive rate of 0.31 at a 1% false positive rate threshold. This indicates reasonably strong performance at detecting spam emails while maintaining a low false positive rate.

To further refine the model, we could ensemble this approach with other types of classifiers to improve the true positive rate. We could also involve end users in the evaluation by deploying the classifier and asking them to label any

emails that were misclassified. This human feedback could be used to re-train the model iteratively.

Overall, the XGBoost classifier provides a solid baseline model for personalized spam filtering that balances accuracy and complexity. With more training data and additional techniques like ensembling, the true positive rate could likely be further improved.

- **Cross-Validation:** The model's performance was assessed using 10-fold cross-validation, employing the area under the receiver operating characteristic curve (AUC) as the evaluation metric.
- **Hyperparameter Tuning:** Grid Search was used to optimize hyperparameters, resulting in the best parameters: `n_estimators`=200, `learning_rate`=0.1, and `max_depth`=4.
- **Model Evaluation:** The model was evaluated on the test set, yielding an AUC score of 0.909, F1 score of 0.804, and accuracy of 85.22%. These metrics indicate the model's effectiveness in distinguishing spam and non-spam emails.

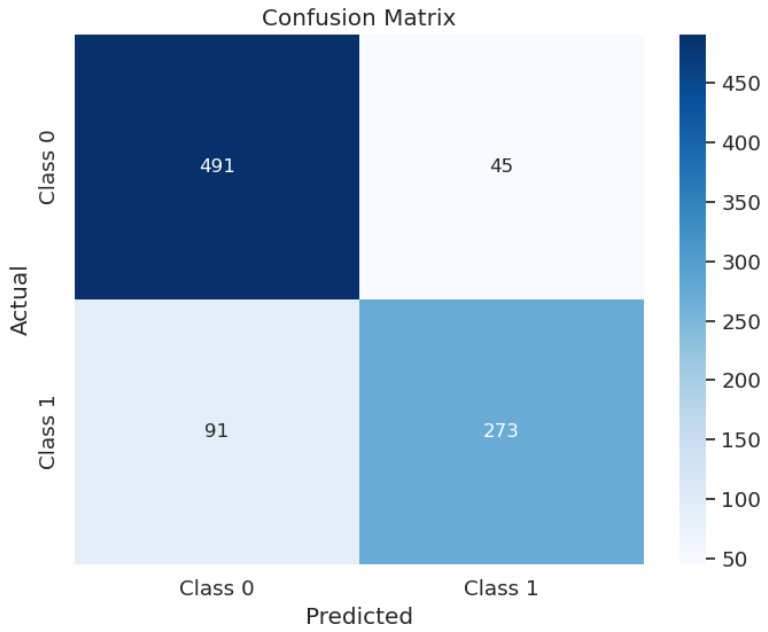

Figure 1: Confusion Matrix

10-fold cross-validation mean AUC	0.9089825434010315
Test set AUC	0.9090187286908599
F1 Score	0.8041237113402061
10-fold cross-validation mean AUC	0.9089825434010315

Table 1: Results

4 Visualizations

Two graphs were generated to visualize the model's performance:

- **Graph 1:** This graph displays the sorted target values of the test set in blue and the predicted values in red, providing a clear visual comparison between actual and predicted values.

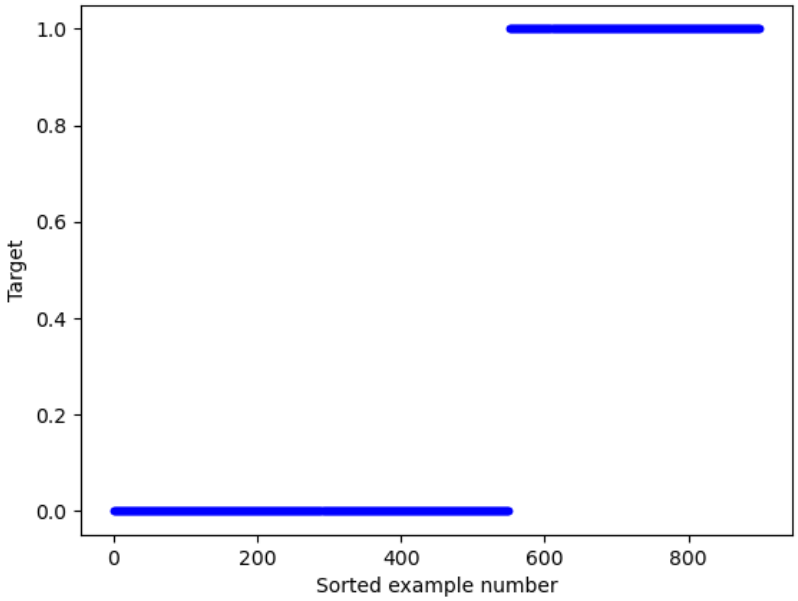

Figure 2: Graph 1: Actual vs Predicted Values

- **Graph 2:** This graph visualizes the output of the model, aiding in the comparison between the sorted example numbers, the target values, and the predicted values.

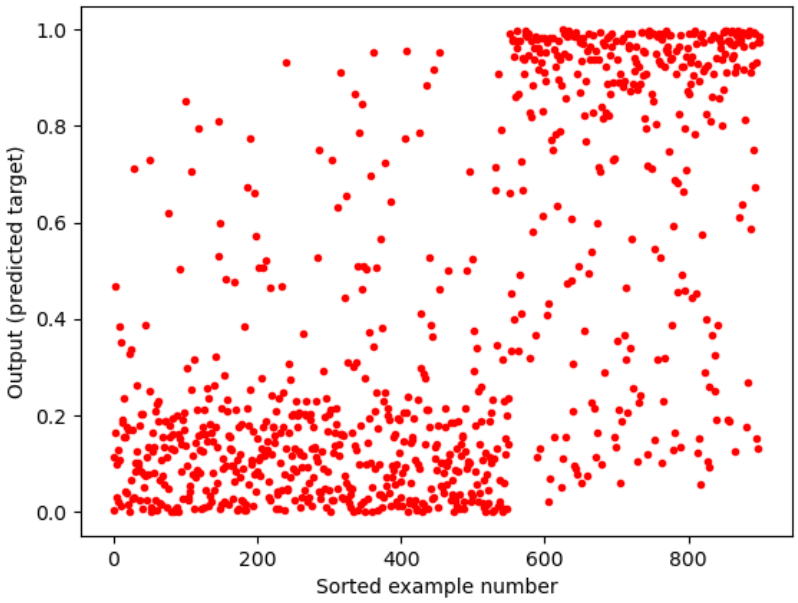

Figure 3: Graph 2: Model Output

In conclusion, the XGBoost algorithm, with its ability to handle missing values seamlessly, proved to be highly effective in classifying spam emails. The comprehensive evaluation metrics and visualizations provided valuable insights into the model's performance, demonstrating its capability to accurately identify spam emails. These results highlight the algorithm's robustness and its potential for real-world applications in email classification tasks.

5 Practical Applications

The XGBoost-based spam email classifier we've developed has diverse practical applications. Email service providers can enhance spam filtering, e-commerce platforms can curate content, and financial institutions can detect phishing emails. Social media networks can improve content moderation, healthcare services enhance data security, and researchers benefit from the model's insights. Marketing teams streamline campaigns, while government agencies protect citizens from scams. This versatile tool strengthens communication, security, and user experiences across various sectors.

Acknowledgements

We would like to express our gratitude to **Professor Kevin Xu** for their valuable guidance throughout the course **CSDS340: Introduction to Machine Learning**.