Classification Methods: Random Forest and BERT-Based Spam Detection

CSC 4850 Machine Learning Project

Deepak Govindarajan

December 5, 2025

1 Introduction

This report describes two machine learning classification approaches implemented for CSC 4850. The first project addresses multi-class classification using Random Forest ensembles across four diverse datasets. The second project implements a binary spam email detector using BERT, a transformer-based deep learning model. Both implementations emphasize adaptive hyperparameter tuning, class imbalance handling, and comprehensive evaluation metrics.

2 Classification: Random Forest Method

2.1 Algorithm Overview

Random Forest combines multiple decision trees through ensemble learning. Each tree trains on a random bootstrap sample of the training data. At each split, only a random subset of features is considered. This approach reduces overfitting and improves generalization compared to single decision trees.

2.2 Dataset Characteristics

Four datasets with varying scales were processed:

- Dataset 1: 150 training samples, 53 test samples, 3,312 features, 5 classes
- Dataset 2: 100 training samples, 74 test samples, 9,182 features, 11 classes
- Dataset 3: 2,547 training samples, 1,092 test samples, 112 features, 9 classes
- Dataset 4: 1,119 training samples, 480 test samples, 11 features, 6 classes

Some datasets contained missing values represented by 1.0×10^{99}, which were handled through mean imputation.

2.3 Preprocessing

Data preprocessing followed three steps. First, missing values were identified and replaced with column means computed from training data. Second, features were standardized using z-score normalization: $(x - \mu)/\sigma$, where statistics were derived from training data only. Third, the same imputation and normalization parameters were applied to test data to prevent data leakage.

2.4 Adaptive Hyperparameter Tuning

Hyperparameters adapted automatically based on dataset size to prevent overfitting on small datasets while allowing complexity for larger ones.

For small datasets (less than 200 samples), conservative parameters were used: 100-150 trees, maximum depth of 3-8 (based on $\log_2$ of feature count), and higher splitting thresholds ($n_{samples}/15$ for min_samples_split, $n_{samples}/30$ for min_samples_leaf).

Medium datasets (200-1000 samples) used balanced parameters: 200-300 trees, depth 8-15, and moderate thresholds ($n_{samples}/40$ and $n_{samples}/80$).

Large datasets (greater than 1000 samples) employed more complex models: 300-500 trees, depth 15-25, and lower thresholds ($n_{samples}/80$ and $n_{samples}/160$).

Dataset 4 received special treatment due to severe class imbalance. It used balanced_subsample class weighting, unlimited depth (max_depth=None), 300-559 trees, and lower splitting thresholds to capture minority class patterns.

2.5 Class Imbalance Handling

When class imbalance ratios exceeded 3:1, the implementation automatically applied balanced or balanced_subsample class weights. This adjustment penalized misclassification of minority classes more heavily during training.

2.6 Implementation Details

The RandomForestClassifier from scikit-learn was used with the following key parameters: bootstrap=True for bagging, oob_score=True for out-of-bag validation, random_state=42 for reproducibility, and max_features set to 'sqrt' or 'log2' based on feature count.

3 Spam Email Detection: BERT Method

3.1 Model Architecture

BERT (Bidirectional Encoder Representations from Transformers) uses a transformer architecture with self-attention mechanisms. Unlike unidirectional models, BERT processes text bidirectionally, capturing context from both directions simultaneously. The implementation used bert-base-uncased (110M parameters) with a binary classification head for ham (0) versus spam (1) classification.

3.2 Dataset Information

The training data consisted of 4,296 labeled emails from two CSV files (spam_train1.csv and spam_train2.csv), with 3,367 ham and 929 spam emails (approximately 3.6:1 ratio). A stratified 80-20 split created training and validation sets, resulting in 3,434 training samples and 859 validation samples. The test set contained 6,448 unlabeled emails.

3.3 Text Preprocessing

Text preprocessing involved multiple steps. Emails were converted to lowercase for consistency. HTML tags were removed using regex patterns. Special characters were eliminated, keeping only alphanumeric characters and spaces. Stopwords were removed to focus on meaningful content. Whitespace was normalized, and empty texts were filtered out. Finally, text labels were encoded numerically: ham=0, spam=1.

3.4 Tokenization and Input Preparation

Text was tokenized using BertTokenizer from Hugging Face Transformers. The tokenizer converted cleaned text into subword tokens, handling out-of-vocabulary words effectively. Special tokens [CLS] and [SEP] were added for classification and sentence separation. Sequences were truncated or padded to a maximum length of 128 tokens. Attention masks distinguished real tokens from padding, ensuring the model ignored padding during computation.

3.5 Training Configuration

Training used the following hyperparameters: learning rate of 2×10^{-5}, 5 epochs with early stopping based on validation F1 score, batch size of 8 (optimized for GPU memory), AdamW optimizer with linear learning rate scheduling, and cross-entropy loss with balanced class weights computed using sklearn's compute_class_weight.

3.6 Class Imbalance Handling

Class imbalance was addressed through weighted loss functions. Balanced class weights were automatically computed and applied to the cross-entropy loss, penalizing misclassification of the minority spam class more heavily. The stratified train-validation split maintained class distribution proportions.

4 Evaluation and Results

4.1 Random Forest Evaluation

The Random Forest implementation generated predictions for all four test datasets. Evaluation metrics included cross-validation accuracy, training accuracy, precision, recall, F1-score, and ROC-AUC scores. Confusion matrices visualized classification performance. Feature importance rankings identified the top 10 most influential features for each dataset.

4.2 BERT Evaluation

The BERT model achieved strong performance on the validation set: accuracy of 98.25%, precision of 96.72%, recall of 95.16%, F1 score of 95.93%, and ROC-AUC of 99.50%. The model converged within 5 epochs, with the best model selected based on highest validation F1 score. Test predictions were generated for all 6,448 test emails, with visualizations comparing validation and test distributions.

5 Conclusion

Both methods successfully addressed their respective classification tasks. Random Forest provided robust multi-class classification with adaptive hyperparameter tuning across diverse dataset scales. BERT achieved excellent binary classification performance for spam detection, leveraging pre-trained language understanding capabilities. Both implementations emphasized proper preprocessing, class imbalance handling, and comprehensive evaluation to ensure reliable and interpretable results.

References

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[2] Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32.

[3] Scikit-learn: Machine Learning in Python. `https://scikit-learn.org/stable/`

[4] Scikit-learn Random Forest Classifier. `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[5] GeeksForGeeks - Machine Learning Resources for Random Forest, Decision Trees and Gini Impurity. `https://www.geeksforgeeks.org/machine-learning/`

[6] Hugging Face Transformers. `https://huggingface.co/transformers/`

[7] PyTorch Documentation. `https://pytorch.org/docs/stable/index.html`

[8] Guide to Tokenization and Padding with BERT: Transforming Text into Machine Readable Data. `https://medium.com/@piyushkashyap045/guide-to-tokenization-and-padding-with-bert-transforming-text-into-machine-readable-data-5`