The Binary Classification of Spam

# GitHub URL

<https://github.com/Crowmium1/UCDPA-L.J._Fitzgerald.git>

# Abstract

This report delves into the Spambase dataset to classify emails as spam or not. A rigorous feature engineering process refined the dataset, with descriptive statistics guiding transformations and feature selections. Four models were used and hyperparameter-tuned for optimal performance. Inferential statistics and hypothesis testing provided deeper insights into the relative efficiencies of these models. Furthermore, ensemble techniques were explored, culminating in a stacking classifier that aimed to harness the strengths of all individual models.

# Introduction

This paper presents a comprehensive evaluation of various machine learning models in their ability to classify emails as 'spam' or 'not spam'. The study evaluates multiple models and aims to determine which model provides the most reliable and accurate email classification using accuracy as the chosen performance metric. Other diagnostic metrics, such as precision, F1 scores, and confusion matrices are addressed.

Email classification is a pivotal task to filter out unwanted messages and prioritize genuine communication. With the continuous growth of electronic mail, automated systems need to be adept at correctly classifying 'spam' and 'not spam' emails.

The machine learning models being evaluated are as follows:

* **K-Nearest Neighbors (KNN)**: A non-parametric method that classifies based on the majority label of its nearest data points.
* **Support Vector Machine (SVM)**: A linear classifier that determines an optimal hyperplane to segregate classes.
* **Random Forest**: An ensemble of decision trees that considers random subsets of features.
* **Gradient Boosting (GB)**: An ensemble method that builds trees sequentially to correct the errors of its predecessor.
* **Bernoulli Naive Bayes (BNB)**: A probabilistic classifier tailored for Binary/Boolean features.
* **Gaussian Naive Bayes (GNB)**: A probabilistic classifier that assumes features follow a normal distribution.
* **Ada Boost (AB)**: An ensemble method that adjusts weights iteratively.
* **Stacked Classifier**: An ensemble approach combining multiple models.

# Dataset

For our study, we sourced a dataset containing labelled emails. Non-spam email origins are from a person’s work and personal sources. There are 58 feature-labelled columns with 4601 rows. 1 column is the ‘spam/ham’ target variable, where 1 is spam and 0 is not spam. The other 57 columns consist of word and character count frequency information. This information is taken from the ‘Spambase’ dataset, found on the UCI repository (see references). There are missing values in this dataset.

# Implementation Process

**Data Import and Preparation**:

1. Modules are imported and functions are kept at the start of the project.
2. Data is read and combined from the datasets into numpy arrays. Columns are checked for consistency.
3. Conducted descriptive statistics and visualized the spam/ham class distribution.
4. Data structures, duplicates, and missing values were managed.
5. Explored the distribution of duplicated rows relative to the target variable.
6. Visualized data correlation through heatmaps and identified high-correlation features.

**Feature Engineering**:

1. Employed principal component analysis (PCA) on the dataset for dimensionality reduction.
2. Developed a dataframe for correlated feature labels. Utilized Random Forest and Gradient Boosting classifiers to derive feature importance values, followed by conversions to dataframe format.
3. Visualized performance metrics of the model against the number of features used, along with ROC curves and reduced dimensionality accordingly.
4. Conducted descriptive statistics on each feature-engineered dataframe.

**Model Evaluation and Hyper-tuning**:

1. Evaluated base model performance using default parameters, ensuring SVM and KNN models were scaled appropriately.
2. Split the data into training and test subsets, applying cross-validation.
3. Implemented loops to train, fit, and predict data, documenting accuracy scores.
4. Generated ROC curves on base data and calculated confidence intervals for accuracy.
5. Iterated the process with hyper-tuned parameters and stored the best models for ensemble training and stacking.
6. Evaluated performance of KNN base and hyper-tuned models using hypothesis testing.

**Ensemble Methods**:

1. Re-split the data and evaluated models on validation data, storing accuracies for hypothesis testing.
2. Printed confusion matrices and classification reports for each model trained on optimal parameters.
3. Visualized model performances and reported validation set scores.
4. Conducted bootstrap confidence intervals on data and formed three null hypotheses concerning model performances.
5. Undertook paired T-tests to determine significant performance differences between models.

**Stacking**:

1. Formed a null hypothesis regarding the stacked model's performance relative to individual ensemble models.
2. Used ensemble models as estimators and instantiated a stacking classifier.
3. Trained and evaluated data on the validation set.
4. Performed a paired T-test comparing the stacked model with the best-performing ensemble method.

**Analysis Plots**:

1. Diagnosed model performance, identifying underfitting, overfitting, or potential data needs with learning curve plots.
2. Employed calibration curves to inspect prediction probability alignment.

**Image Extraction and File Handling**:

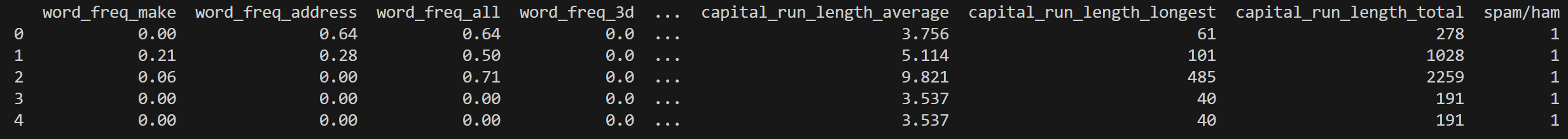
1. Archived the trained model for future access.
2. Transformed the script into an HTML format for improved accessibility.
3. Extracted embedded script images for independent storage.

# Results

**The descriptive statistics:** The original Dataframe gave an indication for the size, shape, and frequency of the data, and the distribution of the target class (fig. 1-3). Duplicates' distribution seemed to match the true target variable distribution (fig. 4); therefore, I believe these emails were not explicitly spam.

**A screenshot of a computer program

Description automatically generated****Fig.1: Spambase descriptive statistics**

**  
Fig.2: Spambase DataframeA graph of ham and spam emails

Description automatically generated  
Fig. 3: Distribution of target class variable in original dataframe.**

**A white sheet with black text

Description automatically generated  
Fig. 4: Distribution of duplicate values vs. target class**

For dimensionality reduction, a correlation matrix (fig.5) and principal component analysis (fig.6) were applied to the Spambase dataset. Highly correlated features with the target variable were deemed valuable. 49 of the original dataset's feature labels comprised 95% of the explained variance.

Out of the three engineered feature variables, the random forest classified feature importance’s showed the best performance on ROC AUCs for the validation set and had the highest TPR and lowest FPR (fig.7). Comparisons between random forest and gradient boosting importance values revealed similarities, with random forest having a lower standard deviation (fig.8-10). Consequently, 30 features were selected for the random forest feature variable for further analysis.

**A blue and red graph

Description automatically generated  
Fig. 5: Correlated feature labelsA graph of a number of different components

Description automatically generated  
Fig. 6: PCA on original dataset**

**A graph of a graph

Description automatically generated with medium confidenceA graph of a graph

Description automatically generated with medium confidenceA graph of a graph of a graph

Description automatically generated with medium confidenceA graph of a graph

Description automatically generated with medium confidence  
A group of graphs showing different colors

Description automatically generated  
Fig. 7: Performance vs. number of features and ROC curves for feature variables.**

**A screenshot of a graph

Description automatically generatedFig.8: Feature Importances  
  
A close up of text

Description automatically generated  
Fig. 9: Feature Variables  
A screenshot of a computer

Description automatically generated  
Fig.10: Descriptive Statistics for feature variables.**

**The inferential statistics:** Four base models were evaluated, and the SVM model outperformed in terms of accuracy, precision, recall, and F1 score. The confidence intervals indicate results falling in that range 95% of the time (fig.12).

After hyper-tuning, SVM and KNN showed slight accuracy improvements (fig.13). Hyper-Tuned SVM exhibited high specificity without compromising sensitivity. GNB and BNB saw no changes after hyper tuning, due to limited tuning of hyperparameters. All models were likely quite well-tuned already, as the baseline performance was already very high.

The ROC AUC curves (fig.14) between base and hyper-tuned models show negligible differences.

***False Positive (FP)*** *or type 1 error or sensitivity: A genuine (not spam) email is classified as spam. This means an important email might get missed or deleted.*

***False Negative (FN)*** *or type 2 error or specificity: A spam email is classified as genuine. This means the user gets unwanted emails in their inbox.*

*The cost of an FP might be higher if a critical email is missed, especially in a business context, while the cost of an FN might be lower since it would mean a minor inconvenience of deleting or marking the email as spam manually. Therefore, in terms of balancing sensitivity and specificity, we can aim for a higher specificity (true negative rate). The cost of missing a genuine email could be higher than the inconvenience of manually sorting out a spam email that lands in the inbox.*

**A group of numbers on a white background

Description automatically generated Fig.11: TPR, FPR, TNR, FNR**

Notably, both optimized KNN and SVM models have slightly improved in correctly identifying 'spam' emails with a higher TPR and not misclassifying 'not spam' emails as 'spam' with a lower FPR (fig.11 and 15). The optimized SVM model performs the best in terms of highest TPR and TNR and lowest FPR and FNR.

**Hypothesis 1:** The obtained p-value comparing hyper-tuned KNN to the base KNN was 0.6124, failing to show a significant improvement, where alpha=0.025. In this case, we failed to reject the null hypothesis and there is no statistically significant improvement in performance from the base model to the hyper-tuned model. The t-statistic of -0.5069 and effect size of 0.0119 corroborate this (fig.16).

**A document with numbers and text

Description automatically generated with medium confidence  
Fig. 12: Inferential Statistics for Base Models**

**A screenshot of a document

Description automatically generated  
Fig. 13: Inferential Statistics for hyper-tuned ModelsA graph of a receiver operating characteristic curve

Description automatically generated  
A graph of a receiver operating characteristic curve

Description automatically generated  
Fig. 14: ROC curves for base vs. hyper-tuned models  
A screenshot of a graph

Description automatically generatedA screenshot of a graph

Description automatically generated  
Fig. 15: Confusion matrices**

**A screenshot of a computer program

Description automatically generated  
Fig. 16: Hypothesis 1**

**All ensemble models** had similar accuracy scores of about 93%. Confusion matrices revealed more false negatives than false positives for all models, with Ada boost underperforming in precision. The F1 scores indicated a balanced performance between precision and recall for all models (fig.17 and 18). Paired t-tests comparing model pairs showed no significant performance differences (fig.19).

Observing the confusion matrices, all models have more false negatives than false positives, meaning they are more likely to miss 'spam' emails than to misclassify 'not spam' emails as 'spam'.

The random forest classifier has more false negatives and fewer false positives than AB and GB, meaning it misses a higher proportion of actual spam. This is preferable to important emails going straight to junk (fig. 18).

All models demonstrate a high degree of confidence, with overlapping intervals, indicating comparable performance. This suggests that there isn’t overfitting, as the data is generalizing well to new data (fig.18).

The p-values from paired t-tests reveal no significant difference in performance between any of the model pairs (AB vs RF, AB vs GB, or RF vs GB). The high p-values (>0.05) lead to failing to reject the null hypotheses. The effect sizes, which quantify the magnitude of the difference between models, are near zero, reiterating the lack of any substantial difference in performances (fig.19). **A graph with a number of squares

Description automatically generated with medium confidence  
Fig. 17: Ensemble model performanceA screenshot of a document

Description automatically generatedFig.18: Ensemble Model Inferential Statistics**

**A screenshot of a computer

Description automatically generated  
Fig.19: Hypothesis 2**

*Learning curves are plotted for the entire dataset, while calibration and ROC curves target the validation set. These curves diagnose model bias and variance, providing insights into potential overfitting or underfitting.*

The random forest classifier is showing slight overfitting as the test curves fail to fully converge with the training curves and there is a fair bit of variation between them after they plateau. This may be due to model complexity and suggests the model doesn't generalize well to unseen data. Simply adding more data is unlikely to enhance the model's performance. Instead, increasing the model's complexity by introducing new richer features may offer improvement. Ada boost and gradient boosting classifiers are well-balanced (fig. 20).

**A graph of a graph

Description automatically generated with medium confidence  
Fig.20: Ensemble models learning curves.**

Calibration curves are essential for gauging actual probability scores. The more aligned a model's forecast is to the diagonal line, the better its calibration. Among all ensemble models, the gradient boosting classifier aligns best with this diagonal line. The probability estimate is not very useful for this use case, where we are more interested in the binary classification of the model, measured by accuracy (fig.21).

**A graph of a graph

Description automatically generated with medium confidence  
Fig. 21: Ensemble models’ calibration curves.**

**For Stacking** an accuracy of 0.9403 was obtained, which is higher than the RF stands at 0.9349. A second hypothesis test was formed to test the performance difference between the stacked model and the best-performing model.

The ROC curve portrays minimal difference between the stacking and gradient boosting classifier (fig.23). Both models perform exceptionally well in reducing type 1 errors, which is of prime importance for this use case.

A paired t-test obtained p-value was 0.3534, which is greater than the common significance level of 0.05. The consequence of this result was the failure to reject the null hypothesis (H0). This means that there's no statistically significant evidence to suggest differing performance between the stacked model and the RF. The t-statistic of -0.9284 and effect size of 0.0224 backs this decision (fig.22).

Both the gradient boosting and stacked classifiers are well-balanced models (fig.24).

**A screenshot of a computer

Description automatically generated  
Fig.22: Hypothesis 3**

**A graph of a curve

Description automatically generated  
 Fig. 23: ROC curves for best classifier vs. stacking classifier.  
A graph of a training curve

Description automatically generated with medium confidenceA graph of a graph of a graph

Description automatically generated with medium confidence  
Fig. 24: Stacking vs. Gradient Boosting Curves**

**A screenshot of a computer

Description automatically generated  
Fig.25: Hypothesis 3**

# Insights

* Hyper-tuning for KNN did not yield a statistically significant improvement in performance in terms of accuracy.
* All in all, the optimized SVM model performs the best in terms of highest TPR and TNR and lowest FPR and FNR, making it the model of choice for this use case, since avoiding misclassification is a priority.
* Statistical tests reveal no notable differences in performance among all ensemble models studied, but the random forest classifier produces fewer false positives, which is preferable.
* Gradient boosting is the top performer out of the ensemble models with consistent results, RF is similar but with more variation, while Ada boost has good performance and good variability.
* The stacked model has marginally better accuracy than gradient boosting, but the difference isn't statistically significant. Both are well-balanced models and align well with their calibration curves. Choosing between them should extend to aspects like computational efficiency, model complexity, and interpretability.
* Some things which could be changed in using scaled data for the ensemble methods. Reduce the code used for functions by using more in-built methods. Select the training set size

Works Cited

ISLAM, NAIF. “Spambase Dataset.” *Www.kaggle.com*, 21 Jan. 2023, www.kaggle.com/datasets/naifislam/spambase-dataset. Accessed 21 Aug. 2023.

“Multiclass Classification with Xgboost in R.” *Stack Overflow*, 21 Aug. 2021, stackoverflow.com/questions/57824461/multiclass-classification-with-xgboost-in-r. Accessed 21 Aug. 2023.

PABLOV. “Naive Bayes & SVM Spam Filtering.” *Kaggle.com*, 21 Aug. 2015, www.kaggle.com/code/pablovargas/naive-bayes-svm-spam-filtering. Accessed 21 Aug. 2023.

SHARMA, SACHIN. “Spambase Classifier(Acc 95%).” *Kaggle.com*, 21 Aug. 2020, www.kaggle.com/code/sachinsharma1123/spambase-classifier-acc-95. Accessed 21 Aug. 2023.

“Spambase.” *Www.kaggle.com*, 21 Aug. 2018, www.kaggle.com/datasets/monizearabadgi/spambase.

Toushik Wasi, Azmine . “Tabular Playground Series - Aug 2022.” *Kaggle.com*, 21 Aug. 2023, www.kaggle.com/competitions/tabular-playground-series-aug-2022/discussion/341034. Accessed 21 Aug. 2023.

UCI Repository. “UCI Machine Learning Repository.” *Archive.ics.uci.edu*, 30 June 1999, archive.ics.uci.edu/dataset/94/spambase.

*DataCamp*, DataCamp, 21 Aug. 2023, app.datacamp.com/.

# Extra File

Another Python file was created to show some other techniques I learned in this course. Generators, list comprehensions, regular expressions, and web scraping are showcased in this file.

Titles taken from the RTE and BBC news webpages were scraped using the beautiful soup and requests libraries. A function utilizing \*\*kwargs argument was used for this scenario, as after inspecting the HTML elements, RTE and BBC have different naming conventions for their titles.

The generator is commented out, and a list comprehension is used instead because the memory savings from using a generator in this situation would be minimal. The simplicity of using a list comprehension instead is better, although if you want to scrape thousands of titles, then a generator would be preferable.