**Machine Learning Project Report:**

Name: Vu Dao & Sheila Corona

Github: https://github.com/joey-congvu/CSC4850_Project.git

## Part 1: Classification

## Introduction

This project evaluates several supervised machine learning models across four datasets with different dimensionality, class structure, and sample size. The goal is to compare model performance under a consistent preprocessing and evaluation pipeline, then select and train the best classifier for each dataset to generate the final test predictions.

The project compares four models — SVM (Linear), multinomial Logistic Regression, KNN, and Random Forest — using the same evaluation setup for all datasets. I used Stratified 5-Fold Cross-Validation so the class distribution stays consistent across folds, which makes the results more reliable. For metrics, I looked at accuracy, macro precision, macro recall, and macro F1. Macro metrics were necessary because some datasets are imbalanced, and macro averaging forces the evaluation to treat all classes equally instead of letting majority classes dominate the score.

## 1. Data Preprocessing (Shared Across All 4 Datasets)

All datasets went through the same preprocessing pipeline. Missing values were detected using a sentinel threshold (values =1e99) and replaced with NaN, then filled using column-wise mean imputation computed only from the training data to avoid leakage. For models that depend on feature scale—SVM, Logistic Regression, and KNN—I applied standardization through sklearn Pipelines. PCA was used only for Dataset 1 and 2 to retain 95% variance because these datasets had correlated features and benefited from dimensionality reduction. After these steps, each dataset was fully numeric, cleaned, and ready for model training.

## 2. Model Selection and Evaluation with Cross Validation

## Dataset 1

After preprocessing and PCA, I evaluated Logistic Regression, Linear SVM. The PCA-transformed data became almost linearly separable, and the Linear SVM ended up outperforming Logistic Regression in both accuracy and macro metrics. Because Linear SVM produced the best and most stable results across the 5 folds, it was selected as the final model for Dataset 1 and retrained on the full training set to generate test predictions.

### Dataset 2

Similar to Dataset 1, I applied preprocessing and PCA. Here, Logistic Regression performed better than Linear SVM. Logistic Regression showed higher accuracy and more stable macro precision/recall across folds, so I selected it as the best model for Dataset 2 and used it to produce the final test predictions.

### Dataset 3

Dataset 3 is high-dimensional (112 features) with 9 classes. After preprocessing, I evaluated SVM, Logistic Regression, KNN, and Random Forest. Using Stratified 5-Fold CV with macro metrics, Random Forest produced the strongest overall performance. Tree-based models captured the nonlinear patterns better than linear or distance-based approaches. Random Forest was selected as the final model and used to generate the predictions for Dataset 3.

### Dataset 4

Dataset 4 has only 11 features but 6 classes and noticeable class imbalance. After preprocessing, I evaluated SVM, Logistic Regression, KNN, and Random Forest. Accuracy was decent, but macro precision, recall, and F1 were low due to minority classes being difficult to learn. Random Forest achieved the best balance across all metrics and the most consistent performance across folds. It was selected as the final model for Dataset 4 and used for test predictions.

## 3. Hyperparameter Tuning

After selecting the best model for each dataset, I fine-tuned its key hyperparameters to improve predictive accuracy and reduce errors before generating the final test predictions. Light tuning, such as adjusting the number of estimators and tree depth for Random Forest or modifying C for Linear SVM, helped stabilize performance and reduce misclassification, especially for Dataset 3 and 4. Overall, this step showed the importance of tuning models to better match each dataset's structure.

## Conclusion

Across the four datasets, the results show that different data structures benefit from different types of models. For Dataset 1, PCA made the data close to linearly separable, and Linear SVM outperformed Logistic Regression. For Dataset 2, Logistic Regression was stronger and more stable after PCA. Dataset 3 is high-dimensional and nonlinear, making Random Forest the best match. Dataset 4, though low-dimensional, suffers from class imbalance, and Random Forest handled this imbalance better than SVM, Logistic Regression, or KNN. Overall, the project demonstrates how preprocessing, dimensionality, class imbalance, and model selection all significantly affect performance and that no single classifier is universally best across all datasets.

## Part 2: Spam Detection

### 1. Preprocessing

I standardized both training datasets to the same format (text, label), removed useless columns, cleaned the email text (lowercasing, removing HTML, replacing URLs/emails, removing noise), combined the two training sets, and converted all text into TF-IDF features. This produced the final X_train, y_train, and X_test matrices used for modeling.

### 2. Model Selection and Evaluation with Cross Validation

I trained three baseline classifiers to establish initial performance before tuning.

The first model was a Decision Tree using the default gini impurity, no maximum depth limit, and class_weight="balanced" to compensate for spam/ham imbalance.

The second model was a Support Vector Machine with a linear kernel, C=1.0, probability estimates enabled, and class_weight="balanced".

The third model was a simple neural network (MLP) with one hidden layer of 100 units, ReLU activation, the Adam optimizer, alpha=0.1 for regularization, and a maximum of 1000 training iterations.

All three models were evaluated using Stratified 5-Fold Cross-Validation to keep the class distribution consistent across splits. For each model, I calculated accuracy, precision, recall, F1-score, ROC-AUC, and generated a full classification report. This provided a consistent baseline that I later compared to the tuned models.

### 3. Hyperparameter Tuning

After comparing the baseline models, I selected the Support Vector Machine (SVM) for tuning because it achieved the highest F1-score and the strongest recall on the spam class, indicating better overall spam-detection performance than the Decision Tree or the MLP. SVM also performs particularly well on high-dimensional TF-IDF text data and is more stable and efficient to tune.

For the tuning step, I used GridSearchCV with Stratified 5-Fold Cross-Validation and optimized the model based on F1-score. Since this is a linear SVM, the main hyperparameter that controls its margin and regularization strength is C, so I searched over several values of C (0.01, 0.1, 1, 5, 10). The tuning process selected the C value that produced the best F1-score, and this final optimized SVM was used for training on the full dataset and generating the test predictions.

## Conclusion

This project successfully developed an effective spam detection system by combining thorough text preprocessing, TF-IDF feature extraction, and machine learning classification. After evaluating multiple baseline models, the Support Vector Machine emerged as the most reliable classifier, achieving the highest F1-score and the strongest recall for spam emails—both critical for minimizing missed spam. Hyperparameter tuning further improved its performance by optimizing the regularization parameter C.

Using the optimized SVM, I generated final predictions for the test dataset. Overall, the system demonstrates strong generalization, high accuracy, and robust spam-identification ability, making it well-suited for real-world email filtering applications.