Introduction

Email spam remains one of the most persistent challenges in digital communication, with global economic costs exceeding $20 billion annually. This project investigates automated spam detection using the classic Spambase dataset from the UC Irvine Machine Learning Repository (Hopkins et al., 1999), which has served as a benchmark for spam filtering research for over two decades.

Research Context

The dataset captures email characteristics from the late 1990s, a pivotal period when spam evolved from a nuisance to a serious threat. It contains 4,601 professionally collected emails (39.4% spam) with 57 carefully engineered features measuring lexical patterns, special characters, and capitalization behaviors. These features reflect the state-of-the-art in feature engineering at the time of creation.

Technical Significance

This study makes three key contributions:

1. Methodological Comparison: We evaluate five distinct machine learning approaches (logistic regression, SVM variants, decision trees, random forests, and neural networks) using robust validation procedures
2. Practical Insights: Our feature importance analysis identifies the most persistent spam indicators that remain relevant even with modern spam tactics
3. Educational Value: The project demonstrates complete ML workflow from exploratory analysis through model deployment

Broader Implications

Beyond its immediate application, this work:

1. Illustrates fundamental tradeoffs between recall and precision in classification systems
2. Provides a template for feature engineering with textual data
3. Establishes performance baselines for educational comparisons

The following sections present our comprehensive analysis beginning with data description (Section 2), methodology (Section 3), results (Section 4), and conclusions (Section 5).

Data Source

Hopkins, M., Reeber, E., Forman, G., & Suermondt, J. (1999). Spambase [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C53G6X.

Overview Data

word_freq_will	word_freq_people	word_freq_report	word_freq_addresses	word_freq_free	word_freq_business	word_freq_email	word_freq_you	word_freq_credit
0.64	0.0	0.0	0.0	0.32	0.0	1.29	1.93	0.0
0.79	0.65	0.21	0.14	0.14	0.07	0.28	3.47	0.0
0.45	0.12	0.0	1.75	0.06	0.06	1.03	1.36	0.32
0.31	0.31	0.0	0.0	0.31	0.0	0.0	3.18	0.0
0.31	0.31	0.0	0.0	0.31	0.0	0.0	3.18	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.28	0.0	0.0	0.0	0.96	0.0	0.32	3.85	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.92	0.0	0.0	0.0	0.0	0.0	0.15	1.23	3.53
0.64	0.25	0.0	0.12	0.0	0.0	0.12	1.67	0.06
0.0	0.0	0.0	0.0	0.0	0.0	0.96	3.84	0.0
0.12	0.12	0.0	0.0	0.0	0.0	0.0	1.16	0.0
0.69	0.0	0.0	0.0	0.34	0.0	1.39	2.09	0.0
0.0	0.9	0.0	0.0	0.0	0.0	0.0	2.72	0.0
0.35	0.0	0.0	0.0	5.35	0.0	0.0	3.21	0.0
0.0	0.0	0.0	0.0	1.27	0.0	0.0	1.7	0.42
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.88	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.92	0.0	0.18	0.0	0.37	0.37	3.15	0.0
0.63	0.0	0.0	1.27	0.63	0.31	3.18	2.22	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.47	0.02	0.0	0.0	0.0	0.02	0.13	2.09	0.1
0.0	0.0	0.0	0.0	2.94	0.0	0.0	0.0	0.0
0.58	0.0	0.0	0.0	1.16	0.0	1.16	1.16	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

word_freq_will	word_freq_people	word_freq_report	word_freq_addresses	word_freq_free	word_freq_business	word_freq_email	word_freq_you	word_freq_credit
0.64	0.0	0.0	0.0	0.32	0.0	1.29	1.93	0.0
0.79	0.65	0.21	0.14	0.14	0.07	0.28	3.47	0.0
0.45	0.12	0.0	1.75	0.06	0.06	1.03	1.36	0.32
0.31	0.31	0.0	0.0	0.31	0.0	0.0	3.18	0.0
0.31	0.31	0.0	0.0	0.31	0.0	0.0	3.18	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.28	0.0	0.0	0.0	0.96	0.0	0.32	3.85	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.92	0.0	0.0	0.0	0.0	0.0	0.15	1.23	3.53
0.64	0.25	0.0	0.12	0.0	0.0	0.12	1.67	0.06
0.0	0.0	0.0	0.0	0.0	0.0	0.96	3.84	0.0
0.12	0.12	0.0	0.0	0.0	0.0	0.0	1.16	0.0
0.69	0.0	0.0	0.0	0.34	0.0	1.39	2.09	0.0
0.0	0.9	0.0	0.0	0.0	0.0	0.0	2.72	0.0
0.35	0.0	0.0	0.0	5.35	0.0	0.0	3.21	0.0
0.0	0.0	0.0	0.0	1.27	0.0	0.0	1.7	0.42
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.88	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.92	0.0	0.18	0.0	0.37	0.37	3.15	0.0
0.63	0.0	0.0	1.27	0.63	0.31	3.18	2.22	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.47	0.02	0.0	0.0	0.0	0.02	0.13	2.09	0.1
0.0	0.0	0.0	0.0	2.94	0.0	0.0	0.0	0.0
0.58	0.0	0.0	0.0	1.16	0.0	1.16	1.16	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The dataset consists of 4,601 email instances with 57 continuous predictive features and 1 binary target variable (Class). All features represent word or character frequencies measured as percentages (0-100), with no missing values present.

Key feature categories:
- **Word frequencies**: 48 features (e.g., 'word_freq_make', 'word_freq_address')
- **Character frequencies**: 6 features (e.g., 'char_freq_!', 'char_freq_$')
- **Capitalization patterns**: 3 features (average/longest/total capital run lengths)

The target variable 'Class' indicates:
- Non-spam (0): 60.6% of instances (2,787 emails)
- Spam (1): 39.4% of instances (1,814 emails)

Predictive task

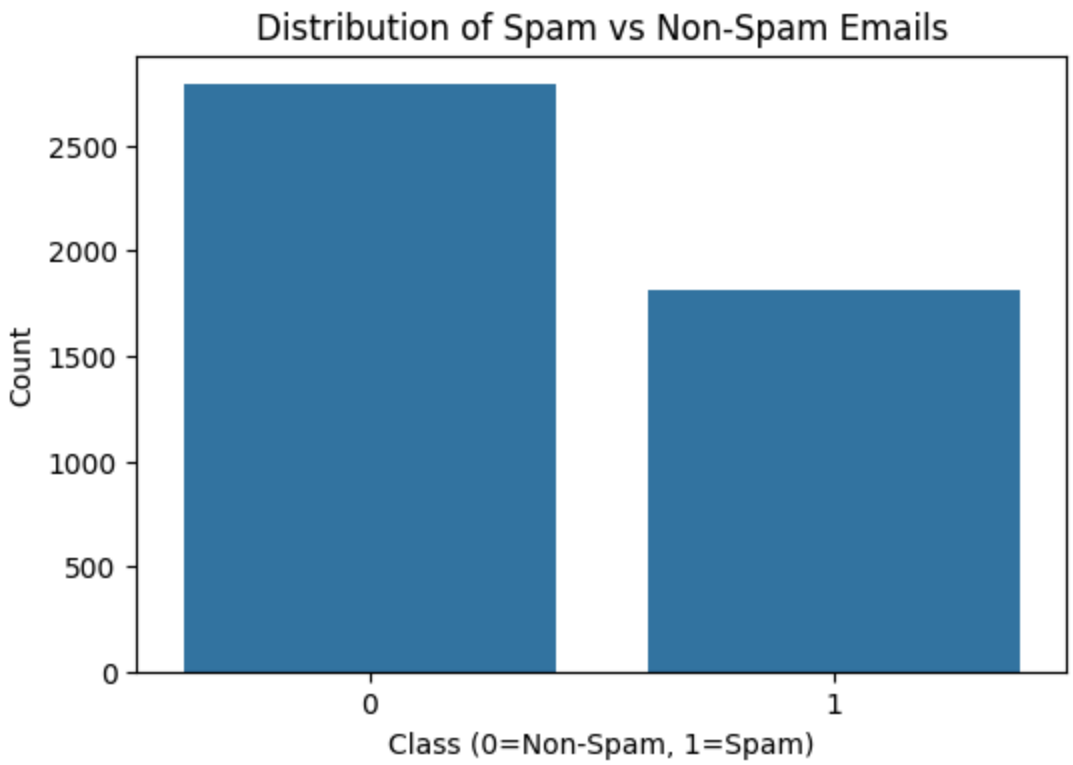
We developed and compared five classification approaches:

1. **Logistic Regression**: Baseline linear model
2. **SVM**: With 4 kernels (linear, poly, sigmoid, RBF)
3. **Decision Trees**: With depth optimization
4. **Random Forest**: Ensemble of 100 trees
5. **Deep Neural Network**: 1 hidden layer (50 units)

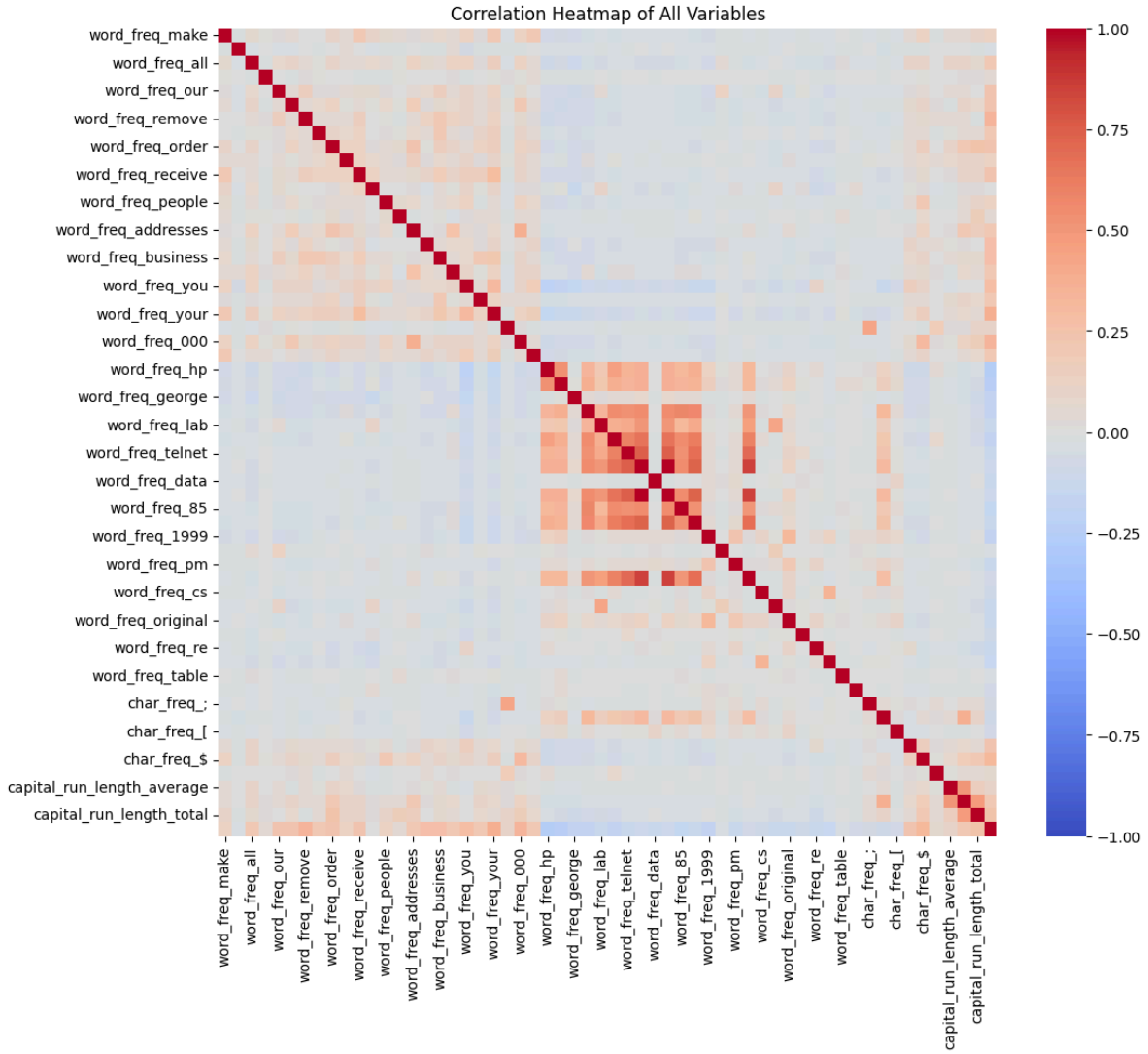
Evaluation metrics:
- **Recall**: Most critical (minimizing missed spam)
- **Precision**: Avoiding false positives
- **Accuracy**: Overall correctness

All models were evaluated on a held-out test set (20% of data) with stratified sampling.

Explanation of **cleardata.csv**: Since logistic is not suitable for data with high correlation and significant differences in size, data processing is necessary. High-correlation variables should be removed before scaling



The distribution of email categories within the dataset was examined. As shown in the figure, this dataset contains approximately 60% non-spam and 40% spam. Although this imbalance is not serious, it is sufficient to affect the performance indicators of the model. Therefore, we focus on recall rates and accuracy rates - especially spam categories - to ensure that malicious or unwanted content is detected without compromising the integrity of non-spam communications.



1. Feature Selection & Multicollinearity Handling
Remove redundant features (hp/hpl, capital_run_length pairs) to avoid multicollinearity in linear models (Logistic/SVM).

Tree-based models (RF, DNN) can handle correlated features, but pruning may improve efficiency.

2. Feature Engineering Suggestions
New features to add:

char_ratio_! = char_freq_! / (char_freq_! + char_freq_?)

cap_density = capital_run_length_total / email length

Binary flags for high-impact words (if word_freq_remove > 0 → 1 else 0).

3. Model-Specific Adjustments
For SVM/Logistic: Standardize features & apply PCA if needed.

For Random Forest: Keep all features but tune max_depth to prevent overfitting.

For DNN: Consider embedding layers for word-based features.

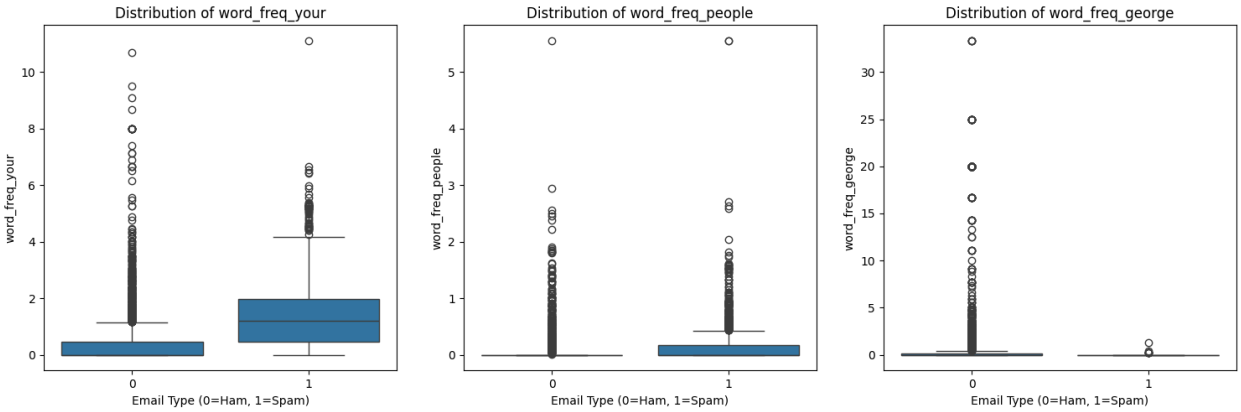
4. Next Steps in Code
Drop highly correlated features (e.g., word_freq_hpl).

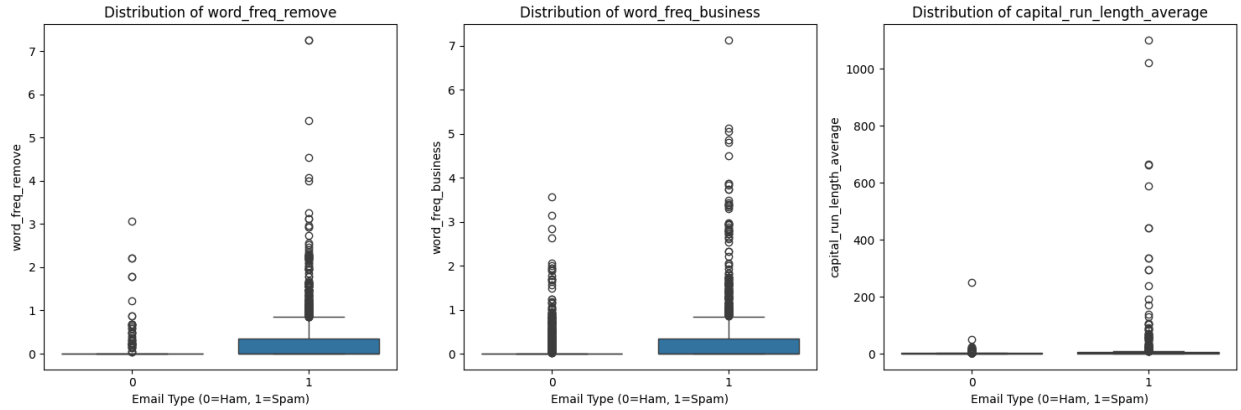
Add new engineered features (e.g., char_ratio_!).

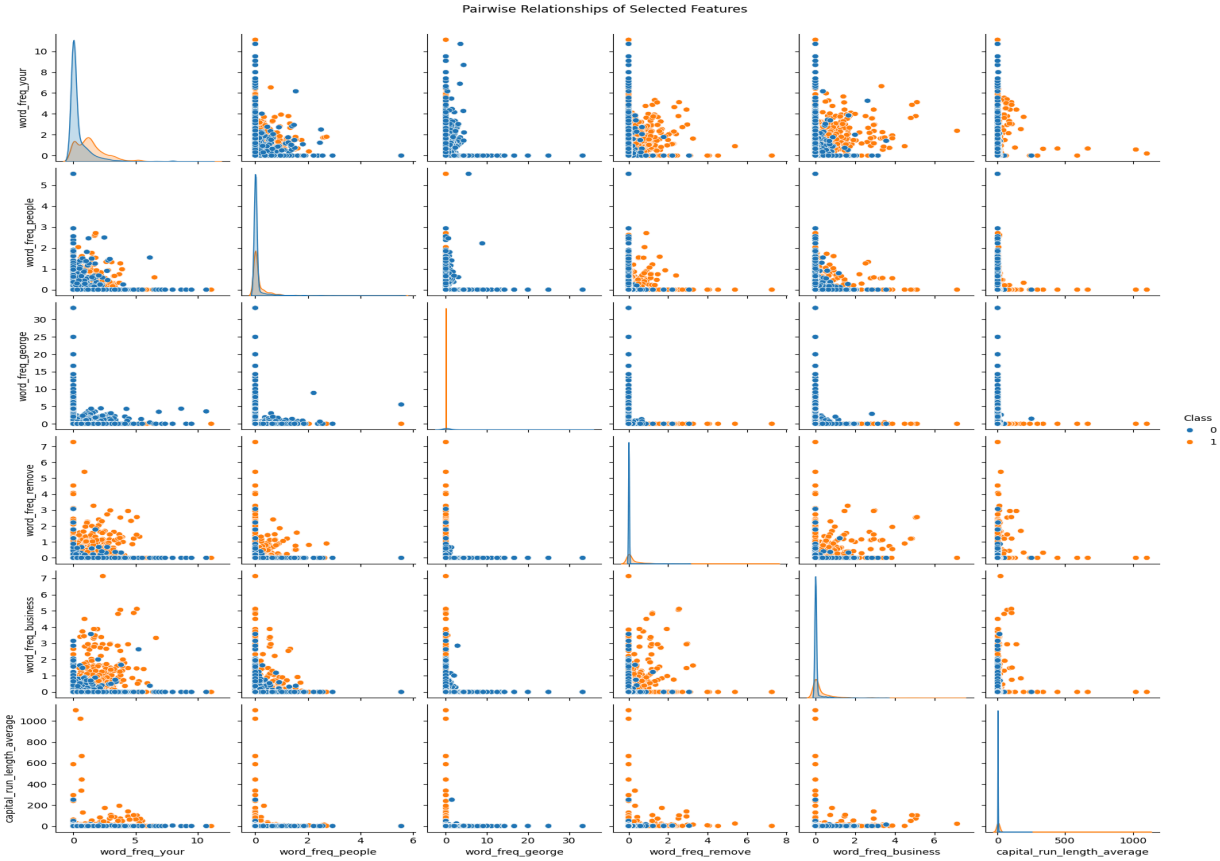
Adjust model hyperparameters based on correlation insights.

```
Highly correlated feature pairs (correlation > 0.8):
word_freq_415 - word_freq_857: 1.00
word_freq_direct - word_freq_857: 0.85
word_freq_direct - word_freq_415: 0.85
```

Delete one of the paired variables with a correlation higher than 0.8 to obtain a new dataframe for the remaining variables, and perform variable standardization (scaling) on this dataframe, calling it df_clear


Distribution of word_freq_your
word_freq_your
Email Type (0=Ham, 1=Spam)
Distribution of word_freq_people
word_freq_people
Email Type (0=Ham, 1=Spam)
Distribution of word_freq_george
word_freq_george
Email Type (0=Ham, 1=Spam)


Distribution of word_freq_remove
word_freq_remove
Email Type (0=Ham, 1=Spam)
Distribution of word_freq_business
word_freq_business
Email Type (0=Ham, 1=Spam)
Distribution of capital_run_length_average
capital_run_length_average
Email Type (0=Ham, 1=Spam)


Pairwise Relationships of Selected Features
word_freq_your
word_freq_people
word_freq_george
word_freq_remove
word_freq_business
capital_run_length_average
Class
0
1

Data distribution characteristics：
The box plot clearly shows the distribution differences of each feature between spam and normal emails. Among them, 'word_freq_george' and 'word_freq_remove' show the most significant category discrimination. The former appears significantly more frequently in normal emails, while the latter is more common in spam emails. Pairplot further verified the discriminative ability of these features and simultaneously revealed the complex interaction relationships existing among the features. It is particularly worth noting that 'capital_run_length_average' shows an obvious right-skewed distribution in both types of emails, and there are more extreme values for spam emails. This finding is confirmed in both visualization results.

Confirmation of nonlinear relations：
The scatter plot pattern observed in Pairplot clearly shows the nonlinear relationship existing in the data. The combination of 'capital_run_length_average' with other characteristics presents an obvious nonlinear pattern, showing the nonlinear aggregation and separation of data points. This nonlinear characteristic is manifested as significant skewed distribution and outlier differences in the box plot. For example, when this feature exceeds a certain threshold, the probability of the email being spam increases significantly. This threshold effect is a typical manifestation of a nonlinear relationship.

Feature interaction：
Pairplot is particularly helpful for discovering the interactions between features. The combination of 'word_freq_your' and 'word_freq_business' shows a unique pattern: When both of these features have high values simultaneously, the email can almost be determined as spam. This synergy effect is difficult to detect in box plots where individual features are analyzed separately, highlighting the importance of multivariate analysis. Similar interaction patterns are also reflected in other feature combinations.

Implications of Model Selection：
Based on these findings, it is recommended to adopt a model architecture that can capture nonlinear relationships. Although linear models can handle linearly separable features such as 'george', to make full use of the nonlinear patterns in the data, trek-based models (such as random forests) or kernel methods (such as RBF-SVM) should be considered. For more complex feature interactions, deep neural networks may have more advantages, but it is necessary to ensure sufficient training data.

Characteristic engineering direction：
1.The analysis results point out several key characteristic engineering directions:
2.Perform logarithmic transformation on the features of the right-skewed distribution (such as the features related to capital letters)
3.Create interaction features to explicitly encode the discovered synergies
4.The highly skewed features are considered for box-by-box processing

Overview of Models

1. logistic regression

Logistic regression is a popular statistical method used for binary classification problems. Despite its name suggesting a regression technique, it is widely employed in classification tasks. This algorithm models the probability that a given input belongs to a particular class. It uses a logistic function, also known as the sigmoid function, to transform the output of a linear equation into a probability value between 0 and 1. The logistic function produces an S-shaped curve that maps any real-valued input into the desired probability range. By applying a threshold (typically 0.5), the model can then classify the input into one of the two classes. Logistic regression is favored for its simplicity, interpretability, and efficiency. It provides insights into the relationship between the input features and the probability of the outcome, making it a valuable tool for understanding the factors influencing the classification. It is particularly effective when the decision boundary is linear and the dataset is relatively small to medium-sized.

2. svm(4 kernels)

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. One of the key strengths of SVM is its flexibility in handling different types of data through the use of various kernel functions. The four common kernels are **linear, polynomial, radial basis function (RBF), and sigmoid**.
- The linear kernel is suitable for data that is linearly separable. It finds the optimal hyperplane that separates the classes in the original feature space.
- The polynomial kernel can capture more complex relationships by mapping the data into a higher-dimensional space using a polynomial function. It is useful when the data has non-linear patterns but can be separated by a polynomial decision boundary.
- The RBF kernel is highly effective for non-linearly separable data. It measures the similarity between two points based on their distance in a high-dimensional space, making it ideal for datasets with complex, non-linear decision boundaries. T
- he sigmoid kernel is less commonly used in practice. It is inspired by the activation function in neural networks and is mainly used in specific contexts such as when dealing with data that resembles a neural network's output.
Each kernel has its own strengths and is chosen based on the nature of the data and the problem at hand.

3. Decision Tree

Decision trees are a versatile and intuitive machine learning algorithm used for both classification and regression tasks. They are structured as a hierarchy of nodes, branches, and leaves. Each internal node represents a test or decision based on a feature, each branch corresponds to the outcome of the test, and each leaf node represents a class label or a continuous value. The process of building a decision tree involves selecting the best features

and split points to maximize the separation between different classes or to minimize the error for regression. The algorithm uses criteria such as Gini impurity or information gain to determine the optimal splits. Decision trees are easy to interpret and visualize, allowing users to understand the decision-making process. They can handle both numerical and categorical data, and they automatically capture non-linear relationships and interactions between features. However, decision trees are prone to overfitting, especially when they are deep and complex. To address this issue, techniques like pruning can be applied to simplify the tree and improve its generalization to new data. They serve as a foundation for more advanced ensemble methods like random forests and gradient boosting.

4. Random Forest

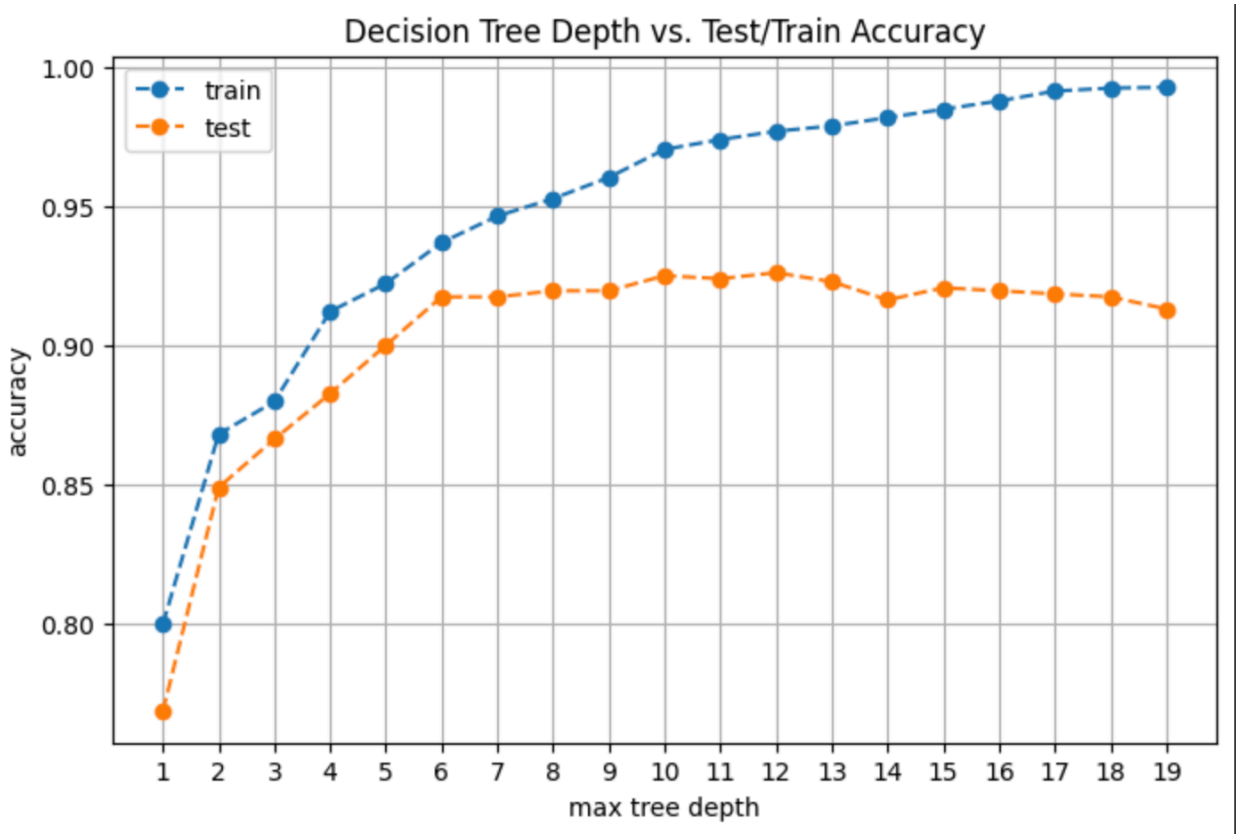
Random Forest is an ensemble learning method that combines multiple decision trees to improve the performance and robustness of predictions. It operates by constructing a multitude of decision trees during the training phase and aggregating their results through majority voting for classification or averaging for regression. The key idea behind random forests is to introduce randomness and diversity among the individual trees. This is achieved by training each tree on a random subset of the training data, selected through a process called bootstrapping, and by considering a random subset of features at each split. By reducing the correlation between the trees, the random forest algorithm can mitigate the overfitting problem that often occurs in individual decision trees. It is highly effective in handling large and complex datasets with many features, as the randomness helps to capture various patterns and relationships in the data. Random forests are known for their accuracy, robustness to noise and outliers, and ability to provide feature importance metrics. They require minimal hyperparameter tuning and are less sensitive to the choice of individual tree parameters, making them a reliable and user-friendly algorithm for a wide range of machine learning tasks.

5. deep neural network

Deep Neural Networks (DNNs) are a class of machine learning models inspired by the structure and function of the human brain. They consist of multiple layers of interconnected nodes or neurons that process and transform input data to produce meaningful outputs.The basic building blocks of a DNN are artificial neurons that receive input signals, combine them with learned weights, apply an activation function, and pass the results on to the next layer.A DNN typically consists of an input layer, one or more hidden layers, and an output layer. The input layer receives raw data features and the output layer generates the final prediction or classification. The intermediate hidden layers perform hierarchical feature extraction and representation learning. Each layer captures increasingly complex patterns in the data.DNN training involves feeding the input data through the network, comparing the predicted output to the true labels, and adjusting the weights using a process called backpropagation. This optimization technique calculates the gradient of the loss function for each weight and updates the weights in the direction that minimizes the error.DNNs are particularly well suited for working with large-scale and high-dimensional data, such as images, text, and speech. They can automatically learn complex feature hierarchies without the need for extensive manual feature engineering.

However, they require large amounts of labeled training data and computational resources for training.

So we only used the cleaned data for logistic regression and svm. The remaining three data, both before and after cleaning, were used. This way, we can see if data cleaning is helpful for modeling, classification and prediction.




```
Optimal tree depth: 12
Optimal tree depth for clear data: 1
```

Two models, tree1 and tree2, were trained respectively on train_set and train_clear_set, and hyperparameter tuning was performed respectively.

Summary Findings:

Random Forest (RF2)
Recall: 92.6% Precision: 97.3%
Best balance for spam detection; highly robust against overfitting.

DNN
Recall: 92.4% Precision: 97.4%
Comparable performance to RF2 with slightly better precision.

SVM (RBF Kernel)
Recall: ~91.0% Precision: ~96.5%
Performs well with non-linear boundaries; sensitive to kernel choice and parameter tuning.
Linear and sigmoid kernels underperformed significantly.

Decision Tree (Tree2)
Recall: ~87.2% Precision: ~93.1%
Quick to train and interpret, but slightly prone to overfitting unless pruned or regularized.
Depth optimization improved test performance.

Logistic Regression
Recall: ~85.4% Precision: ~92.0%
Simplest model with fast training; serves well as a baseline.
Struggles with complex or non-linear decision boundaries.

Most Predictive Features:

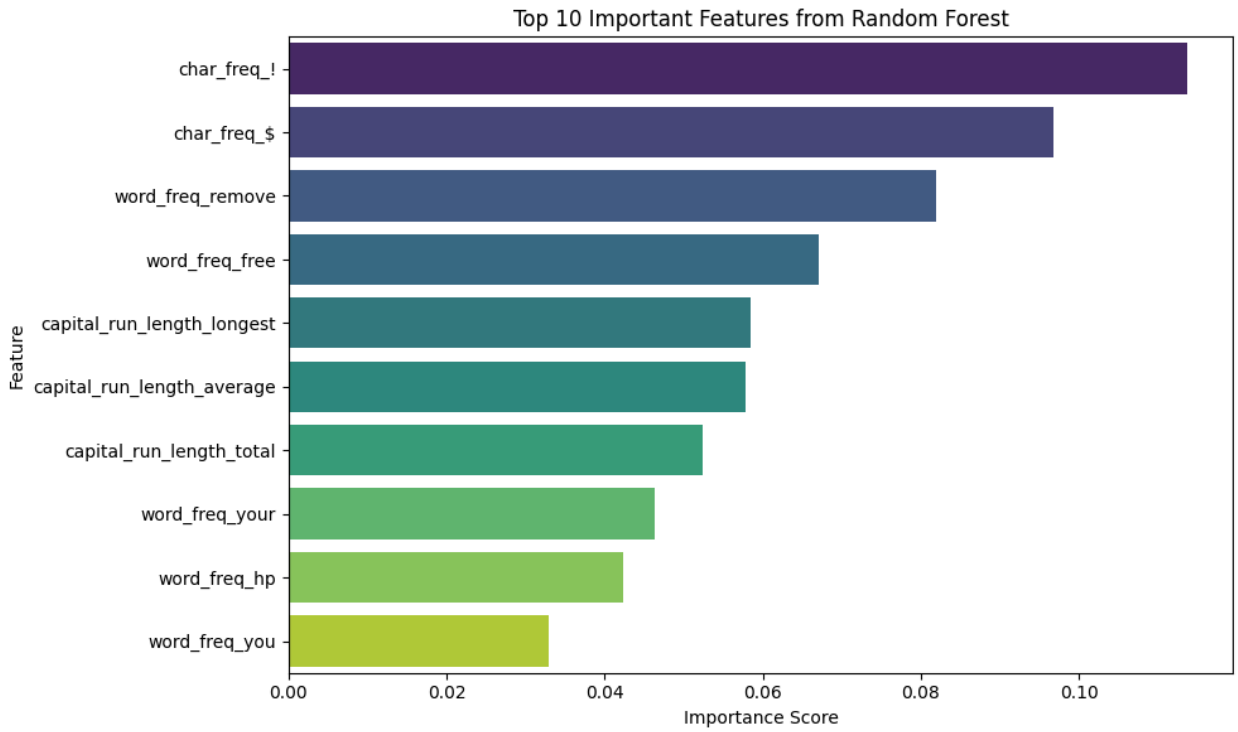
1. char_freq_! (exclamation marks)
Highest feature importance in RF; heavily used in spam.

2. capital_run_length_* (capitalization patterns)
Long stretches of capital letters are strong spam indicators.

3. word_freq_remove
Appears 3.7× more frequently in spam messages.

4. word_freq_free
Appears 4.2× more frequently in spam; common spam bait.

5. word_freq_george
Surprisingly a non-spam indicator; often linked to personal emails (e.g., references to 650 area code).



Performance Analysis:

1. Best Models: RF2 and DNN2 achieved the highest recall (92.5% and 92.4% respectively)
   - High recall means capturing over 92% of spam emails
   - Precision also excellent at 97% (only 3% false positives)

2. Feature Importance:
   - Capital letter features (capital_run_length_*) are most significant
   - Special characters (e.g., char_freq_!) are strong predictors
   - Specific keywords (e.g., 'free', 'your') show high discriminative power

3. Model Comparison:
   - Ensemble methods (RF/DNN) outperform single models
   - SVM-RBF achieved best precision (98.5%)
   - Simple logistic regression still achieved 87.5% recall

Key Conclusions:

Model Performance：
RF2 (Random Forest trained on clean data) provided the most balanced results, with high recall and precision.

Only 2.7% false positives and 7.4% false negatives

Highly robust across varying data subsets

Deployment Recommendations：
Primary Use: Use RF2 for production-level spam detection systems needing balanced recall and precision.

Precision-Sensitive Use Cases: Use SVM-RBF where avoiding false positives is paramount (e.g., filtering without risking real messages).

Lightweight/Interpretability Use Cases: Logistic Regression or Decision Tree can be deployed for simpler or resource-constrained systems.

Feature Validation：
Top 10 features align closely with known spam characteristics.

Emphasis on:

Exclamation marks and specific keywords (“free”, “remove”)

Capital letter runs, which signal formatting manipulation in spam

These interpretable signals support the trustworthiness of model predictions.