International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(Research Article)

Classifying Spams Using Apache Spark MLlib

Mayowa Timothy Adesina [1, *] and Joshua Mayokun Adesina [2]

[1] *College of Business Administration, Kansas State University, Manhattan, KS 66502*

[2] *Sociology Department, Federal University, Oye-Ekiti, Nigeria.*

International Journal of Science and Research Archive, 2024, 12(02), 2094–2112

Publication history: Received on 26 June 2024; revised on 11 August 2024; accepted on 13 August 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.12.2.1432

Abstract

This paper provides a comprehensive overview of various machine learning algorithms, including Logistic Regression, Decision Trees, and Random Forests, with a focus on their application in predictive modeling. The discussion emphasizes the importance of feature selection, engineering, and model evaluation techniques like cross-validation to ensure robust and generalizable models. By leveraging the Spambase dataset from the UCI Machine Learning Repository, the performance of these algorithms is compared and contrasted using key metrics such as accuracy, precision, recall, and F1-score. The paper also highlights the significance of understanding dataset characteristics and feature importance in optimizing model performance. The findings demonstrate that while each algorithm has its strengths and limitations, Random Forests generally provide superior predictive performance, especially in handling complex and high-dimensional datasets. This work serves as a valuable resource for data scientists and researchers looking to understand the practical implications of different machine learning techniques and their impact on real-world data.

Keywords: Machine Learning; Artificial Intelligence; Spam Messages; Logistic Regression; Decision Tree; Random Forest; Apache Spark

1. Introduction

1.1. What is Machine Learning?

1.1.1. Definition

Machine learning is a field of artificial intelligence (AI) that involves the development of algorithms and models that allow computers to learn from and make predictions or decisions based on data without being explicitly programmed to do so. In other words, machine learning is a type of computer programming that enables systems to automatically improve their performance on a task or problem by learning from experience or data.

Machine learning algorithms are designed to identify patterns or relationships within large sets of data, also known as training data. These algorithms are trained on historical or labeled data, and they learn from this data to make predictions or decisions when presented with new, unseen data. Machine learning can be classified into different types, including supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning, among others.

Supervised learning involves training a model using labeled data, where the desired output or target variable is provided along with the input data.

[*] Corresponding author: Mayowa Timothy Adesina

Unsupervised learning, on the other hand, involves training a model on unlabeled data, where the algorithm must identify patterns or relationships within the data without any labeled guidance.

Semi-supervised learning combines elements of both supervised and unsupervised learning, using partially labeled data. Reinforcement learning involves training a model to make decisions or take actions in an environment based on feedback in the form of rewards or punishments.

2. What is machine learning?

2.1. Understanding Dataset

The **“spambase.data”** dataset contains a total of 4,601 email samples, each represented by 57 features extracted from the content and attributes of the email. These features include various characteristics such as the frequency of certain words, the presence of certain characters, and other statistical measures.

The dataset is labeled, with each email sample labeled as either spam (1) or non-spam (0), indicating whether the email is classified as spam or not. The “is_spam” label is the target variable that machine learning models aim to predict based on the features provided.

The dataset will be used for developing and evaluating spam detection algorithms and models. It can be used for binary classification tasks in machine learning to train and test models for predicting spam emails based on the provided features.

2.1.1. Variables in the dataset

word_freq_make: Frequency of the word “make” in the email (continuous).

word_freq_address: Frequency of the word “address” in the email (continuous). word_freq_all: Frequency of all words in the email (continuous). word_freq_3d: Frequency of the word “3d” in the email (continuous). word_freq_our: Frequency of the word “our” in the email (continuous). word_freq_over: Frequency of the word “over” in the email (continuous). word_freq_remove: Frequency of the word “remove” in the email (continuous).

word_freq_internet: Frequency of the word “internet” in the email (continuous).

word_freq_order: Frequency of the word “order” in the email (continuous). word_freq_mail: Frequency of the word “mail” in the email (continuous). word_freq_receive: Frequency of the word “receive” in the email (continuous). word_freq_will: Frequency of the word “will” in the email (continuous). word_freq_people: Frequency of the word “people” in the email (continuous). word_freq_report: Frequency of the word “report” in the email (continuous).

word_freq_addresses: Frequency of the word “addresses” in the email (continuous). word_freq_free: Frequency of the word “free” in the email (continuous).

2.2. Understanding dataset

word_freq_business: Frequency of the word “business” in the email (continuous). word_freq_email: Frequency of the word “email” in the email (continuous). word_freq_you: Frequency of the word “you” in the email (continuous). word_freq_credit: Frequency of the word “credit” in the email (continuous). word_freq_your: Frequency of the word “your” in the email (continuous). word_freq_font: Frequency of the word “font” in the email (continuous). word_freq_000: Frequency of the word “000” in the email (continuous). word_freq_money: Frequency of the word “money” in the email (continuous).

word_freq_hp: Frequency of the word “hp” in the email (continuous). word_freq_hpl: Frequency of the word “hpl” in the email (continuous).

word_freq_george: Frequency of the word “george” in the email (continuous). word_freq_650: Frequency of the word “650” in the email (continuous). word_freq_lab: Frequency of the word “lab” in the email (continuous). word_freq_labs: Frequency of the word “labs” in the email (continuous). word_freq_telnet: Frequency of the word “telnet” in the email (continuous). word_freq_857: Frequency of the word “857” in the email (continuous). word_freq_data: Frequency of the word “data” in the email (continuous).

word_freq_415: Frequency of the word "415" in the email (continuous).

- **others include:** word_freq_85 word_freq_technology word_freq_1999 word_freq_parts word_freq_pm word_freq_direct word_freq_cs word_freq_meeting word_freq_original word_freq_project word_freq_re CHAPTER 1. WHAT IS MACHINE LEARNING?

word_freq_edu word_freq_table word_freq_conference char_freq_semicolon char_freq_leftparen char_freq_leftsquare char_freq_exclamation char_freq_dollar char_freq_hash capital_run_length_average capital_run_length_longest capital_run_length_total

3. Classification Models

3.1. Breakdown of each model Used and Identification of the best model

Three Models were Used for this Project and we will be discussing their utility and outcome.

- Logistic Regression
- Decision Tree
- Random Forest

3.1.1. Logistic Regression

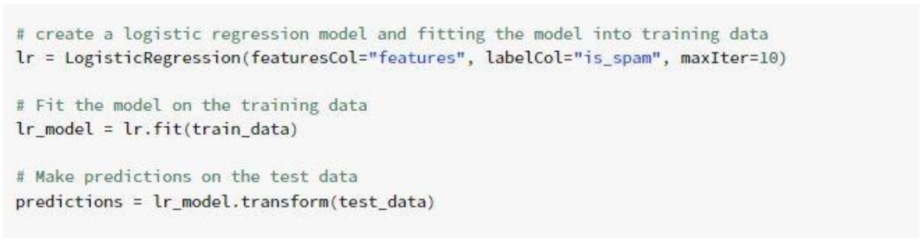
Logistic regression is a type of supervised learning algorithm used for binary classification tasks, where the goal is to predict one of two possible classes. It's commonly used when the target variable is binary (e.g., yes/no, 1/0) and the relationship between the input features and the target is assumed to be linear. Logistic regression uses a logistic function to model the probability of an input belonging to a particular class, and it estimates the parameters that best fit the data during training. Advantages of logistic regression include its simplicity, interpretability, and efficiency in terms of computational resources. It can work well when the relationship between the input features and the target variable is relatively simple and linear. However, it may not be suitable for complex or non-linear relationships.

Analysis of Dataset

After Splitting the dataset using 70/30 training/testing Random splitting and setting seed as 42. The following were done to achieve results that will be discussed later in this chapter.

Creation of Logistic regression model, fitting training dataset and making prediction with testing dataset

knitr::include_graphics("logcrea.JPG")



```
# create a logistic regression model and fitting the model into training data
lr = LogisticRegression(featuresCol="features", labelCol="is_spam", maxIter=10)

# Fit the model on the training data
lr_model = lr.fit(train_data)

# Make predictions on the test data
predictions = lr_model.transform(test_data)
```

Figure 1 Creating Logistic Regression Model

Graph below represent the important features in the Model

knitr::include_graphics("logfea.JPG")

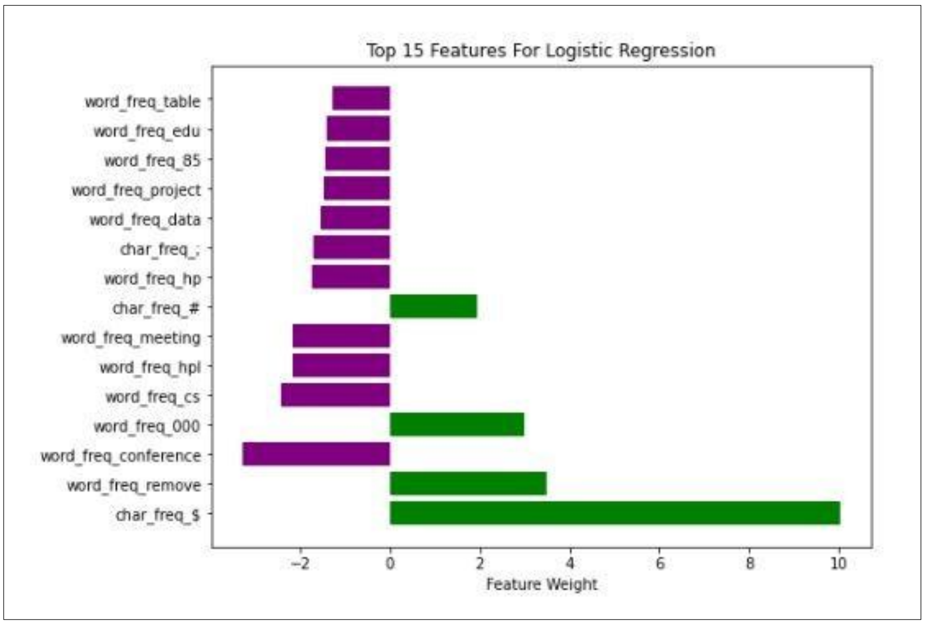
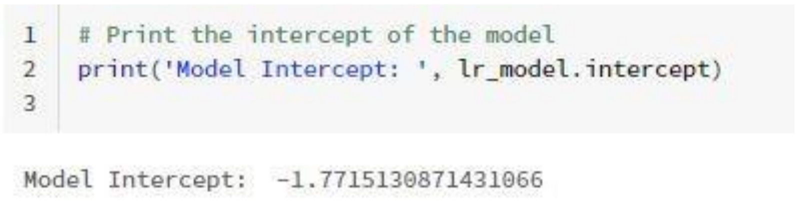


Figure 2 Top 15 Features

-* Char_freq_$ is the top feature in this category, followed by word_freq_remove and word_freq_conference.*

After this, the model intercept was printed knitr::include_graphics("logint.JPG")



```
# Print the intercept of the model
print('Model Intercept: ', lr_model.intercept)

```

```
Model Intercept:  -1.7715130871431066
```

Figure 3 Model Predictability

Reported Model Intercept: -1.7715130871431066. The negative value suggests that the model predicts a lower probability of spam (or a negative outcome) when all input features are set to zero.

knitr::include_graphics("logcurve.JPG")

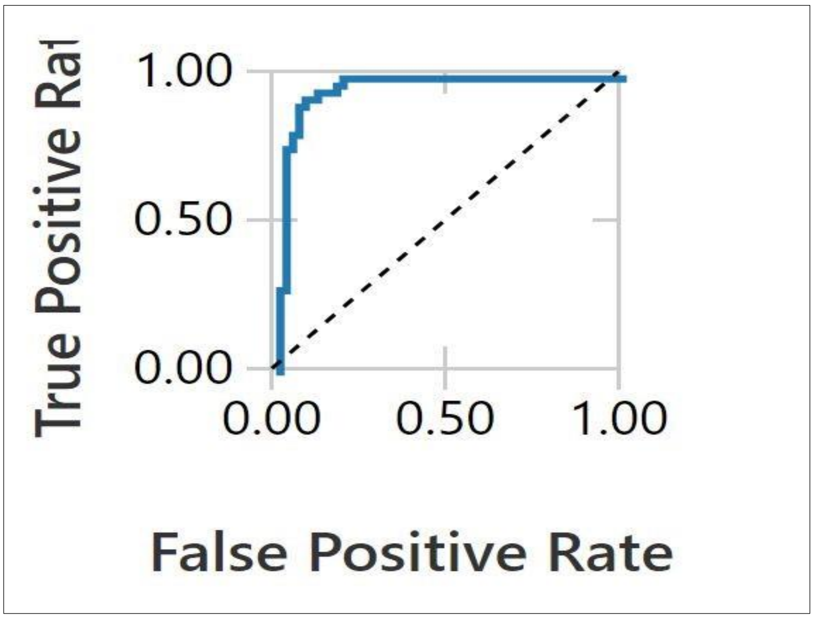


Figure 4 ROC Curve

The "ROC" curve is a graphical representation of the performance of a binary classification model, such as logistic regression, in terms of its true positive rate (TPR) versus its false positive rate (FPR). It is commonly used to evaluate the classification performance of a model and assess its ability to discriminate between positive and negative samples.

3.1.2. Decision Tree

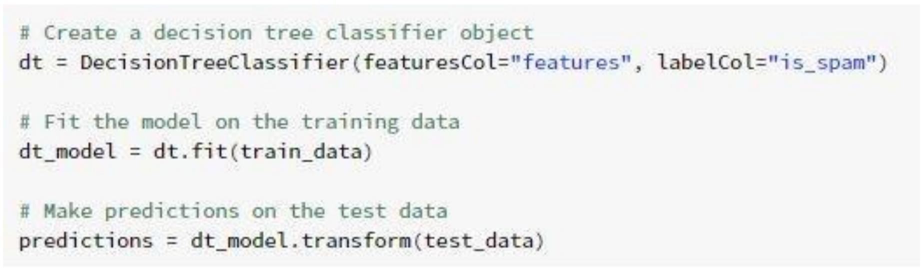
A decision tree is a tree-like structure that is used for both classification and regression tasks. It's a type of supervised learning algorithm that recursively splits the data based on feature values, creating decision nodes and leaf nodes, until a stopping criterion is met. Each decision node represents a decision based on a feature value, and each leaf node represents a predicted class or value. Decision trees are easy to understand and interpret, and they can handle both categorical and numerical features. They are also capable of capturing non-linear relationships between features. However, decision trees are prone to overfitting, as they can become overly complex and memorize the training data, leading to poor generalization performance on unseen data. Techniques such as pruning and setting appropriate hyperparameters can help mitigate overfitting.

Analysis

After Splitting the dataset using 70/30 training/testing Random splitting and setting seed as 42. The following were done to achieve results that will be discussed later in this chapter.

Creation of Decision tree model, fitting training dataset and making prediction with testing dataset

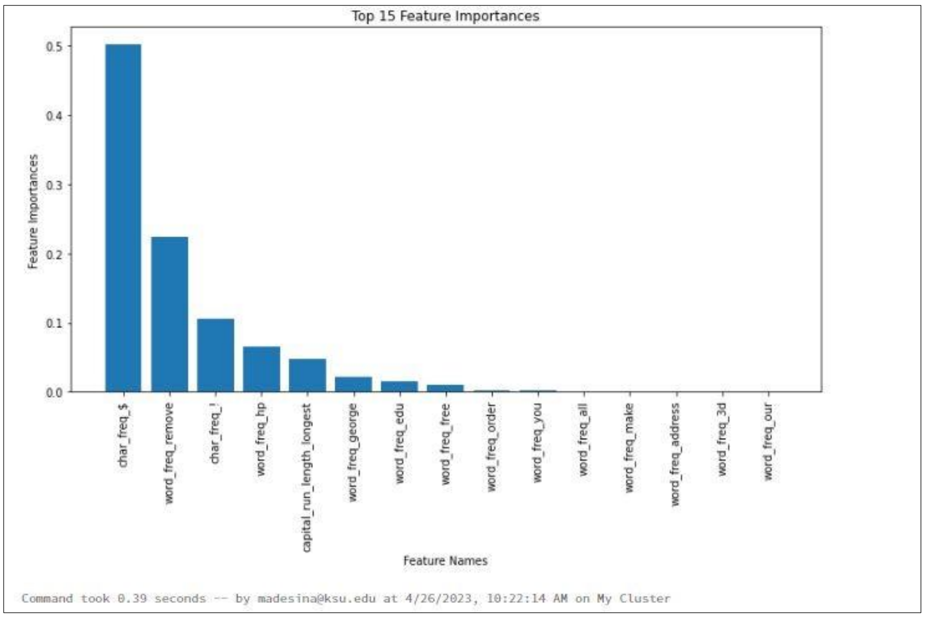
knitr::include_graphics("dtcrea.JPG")



```
# Create a decision tree classifier object
dt = DecisionTreeClassifier(featuresCol="features", labelCol="is_spam")

# Fit the model on the training data
dt_model = dt.fit(train_data)

# Make predictions on the test data
predictions = dt_model.transform(test_data)
```

Figure 5 Creating Decision Tree Model

Graph showing Top Features in Decision tree Model knitr::include_graphics("decfea.JPG")



```
Command took 0.39 seconds -- by madesina@ksu.edu at 4/26/2023, 10:22:14 AM on My Cluster
```

Figure 6 Top 15 Features of the Decision Tree Model

-* Char_freq_$ is the top feature in this category, followed by word_freq_remove and word_freq_!.*

After this, decision tree was printed, please see below, The Decision tree shows how the data was spread starting from Y variable which is the dependent variable.

• The tree has ***37 nodes and 5 depth**, this means that the tree has 37 total nodes, including both internal (decision) nodes and leaf (outcome) nodes, and the longest path from the root node to a leaf node is 5 levels deep. Each level represents a decision based on the value of a specific feature, and the tree grows deeper with each level.

The decision tree is constructed by recursively splitting the data based on the values of the features. At each level, the algorithm selects the best feature to split the data into two or more subsets, typically based on criteria such as Gini impurity or information gain. The process continues until a certain stopping criterion is met, such as reaching the maximum depth, minimum number of samples per leaf, or when no further improvement can be made in splitting the data.

Once the tree is constructed, it can be used for making predictions on new data by following the decision path from the root node to a leaf node based on the values of the features of the input data. The predicted outcome at the leaf node is then used as the final decision or prediction.

knitr::include_graphics("dttree.JPG")

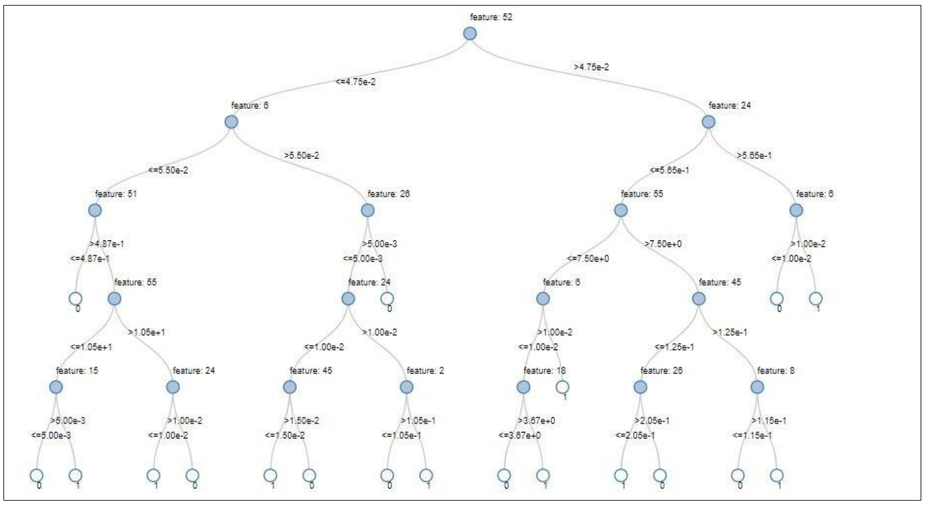


Figure 7 Branches of the Decision Tree

3.1.3. Random Forest

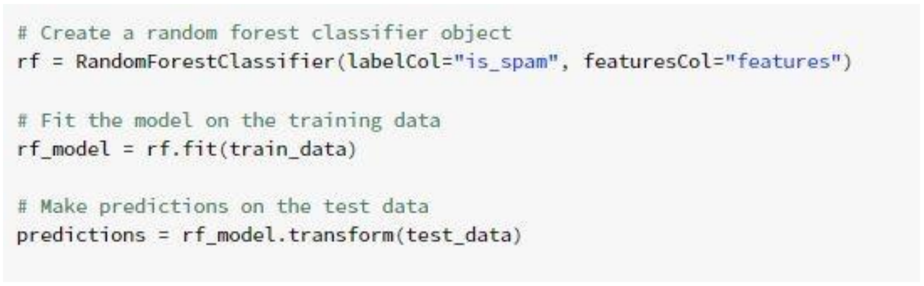
A random forest is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and robustness. It works by constructing a forest of decision trees during training and then averaging their predictions during inference. Each tree in a random forest is trained on a random subset of the data, and a random subset of features is considered at each split, adding randomness to the model. Random forests can overcome some of the limitations of individual decision trees, such as overfitting, by reducing variance and increasing model stability. They can handle large datasets with high-dimensional features and are less susceptible to noise in the data. Random forests are widely used for classification tasks due to their high accuracy and ability to handle complex relationships between features.

Analysis

After Splitting the dataset using 70/30 training/testing Random splitting and setting seed as 42. The following were done to achieve results that will be discussed later in this chapter.

Creation of Random Forest model, fitting training dataset and making prediction with testing dataset

knitr::include_graphics("rfcrea.JPG")



```
# Create a random forest classifier object
rf = RandomForestClassifier(labelCol="is_spam", featuresCol="features")

# Fit the model on the training data
rf_model = rf.fit(train_data)

# Make predictions on the test data
predictions = rf_model.transform(test_data)
```

Figure 8 Creating Random Forest Model

Important features in Ramdom Forest include knitr::include_graphics("ranfea.JPG")

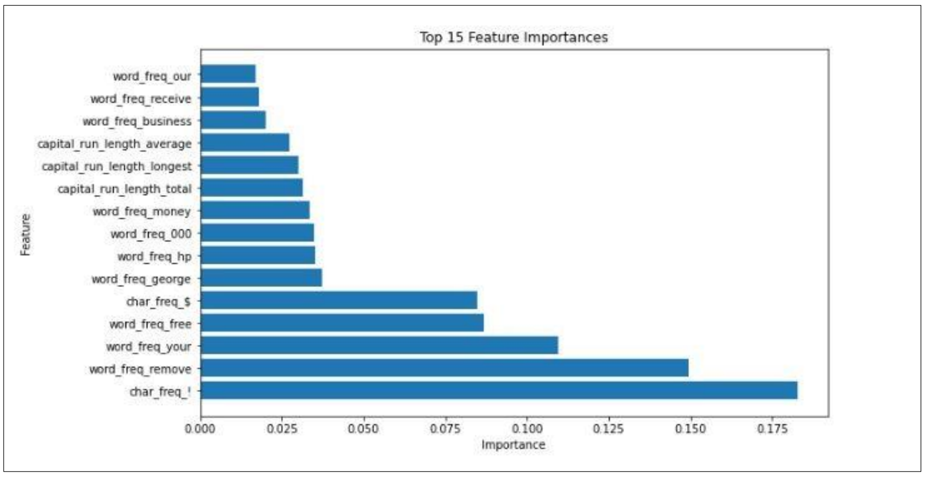


Figure 9 Top 15 Features from the Random Forest Model

-* Char_freq_! is the top feature in this category, followed by word_freq_remove and word_freq_your.*

After this, Receiver Operating Characteristic (ROC) for Random forest was developed. This was achieved by extracting the true positive rate (TPR) and false positive rate (FPR) from the model's predictions on the training and testing data and plotting them on a graph, The plot includes a diagonal line representing the baseline (random) model.Please see below

knitr::include_graphics("rfcurve.JPG")

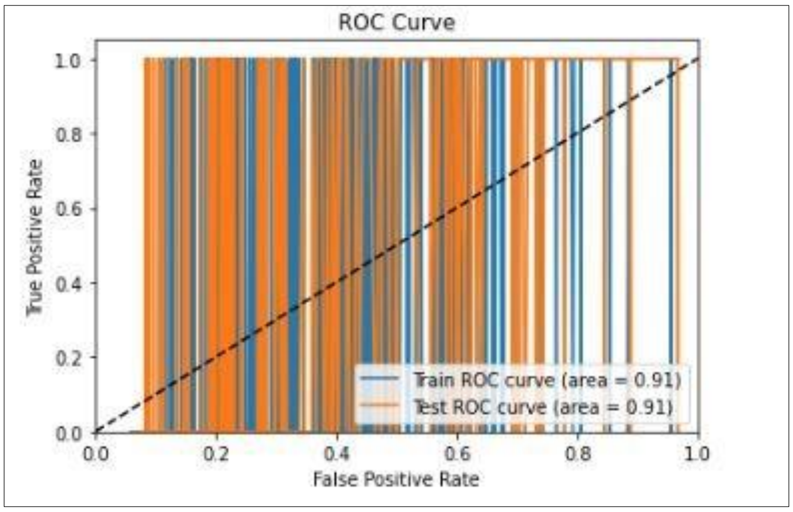


Figure 10 Random Forest ROC Curve

In summary, logistic regression is a simple and interpretable model suitable for linear relationships, decision trees are easy to understand and can capture non-linear relationships, and random forests are an ensemble method that can improve prediction accuracy and handle complex data.

3.1.4. Explanation of Metrics Used to identify the best Model

Area under ROC (Receiver Operating Characteristic) curve: It measures the performance of a binary classification model by plotting the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds. The AUC (Area Under Curve) of the ROC curve is a commonly used metric to assess the model's ability to discriminate between positive and negative samples, with a higher AUC indicating better performance.

Precision: It measures the proportion of true positive predictions out of the total positive predictions made by a model. It represents the accuracy of positive predictions made by the model, and is calculated as Precision = TP / (TP + FP), where TP is the number of true positives and FP is the number of false positives.

Recall (also known as Sensitivity or True Positive Rate): It measures the proportion of true positive predictions out of the total actual positive samples in the data. It represents the model's ability to correctly identify positive samples, and is calculated as Recall = TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives.

F1 score: It is the harmonic mean of precision and recall, and provides a balance between precision and recall. It is a single value that combines

• **Accuracy**: It measures the overall correctness of a model's predictions, and is calculated as the proportion of correctly predicted samples out of the total samples. Accuracy = (TP + TN) / (TP + TN + FP + FN), where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

3.2. Presentation of Results

Here is a breakdown of each score for each model:

Logistic regression:

- Area Under ROC: 0.970
- Precision: 0.918
- Recall: 0.918
- F1 Score: 0.917

Accuracy: 0.918 Decision Tree:

- Area Under ROC: 0.635
- Precision: 0.916
- Recall: 0.912
- F1 Score: 0.911

Accuracy: 0.912 Random Forest:

- Area Under ROC: 0.970
- Precision: 0.923
- Recall: 0.922
- F1 Score: 0.921
- Accuracy: 0.922

3.3. In comparison of the models based on their performance measures

- **Area Under ROC**: Both logistic regression and random forest have the same score of 0.970, while decision tree has a lower score of 0.635. This means that logistic regression and random forest have better discriminatory power than decision tree.
- **Precision**: Random forest has the highest precision of 0.923, followed by logistic regression with 0.918 and decision tree with 0.916. This means that random forest is better at identifying true positives than the other models.
- **Recall**: Both logistic regression and random forest have the same recall score of 0.922, which is higher than decision tree's score of 0.912. This means that logistic regression and random forest are better at identifying true positives than decision tree.
- **F1 Score**: Random forest has the highest F1 score of 0.921, followed by logistic regression with 0.917 and decision tree with 0.911. This means that random forest is better at balancing precision and recall than the other models.
- **Accuracy**: All models have similar accuracy scores, with random forest having the highest score of 0.922 and the other two models having a score of 0.912.

Based on these results, **Random forest is the best model overall** as it has the highest F1 score, precision, and recall, as well as a high area under the ROC curve. Logistic regression also performed well and could be considered as an alternative option. However, decision tree had the lowest scores in all measures and may not be the best option for this dataset.

4. Cross-Validation

4.1. Definition and Analysis

Cross-validation is a technique used in machine learning to assess the performance of a model on unseen data by partitioning the available data into multiple subsets or "folds" and using them for training and validation iteratively.

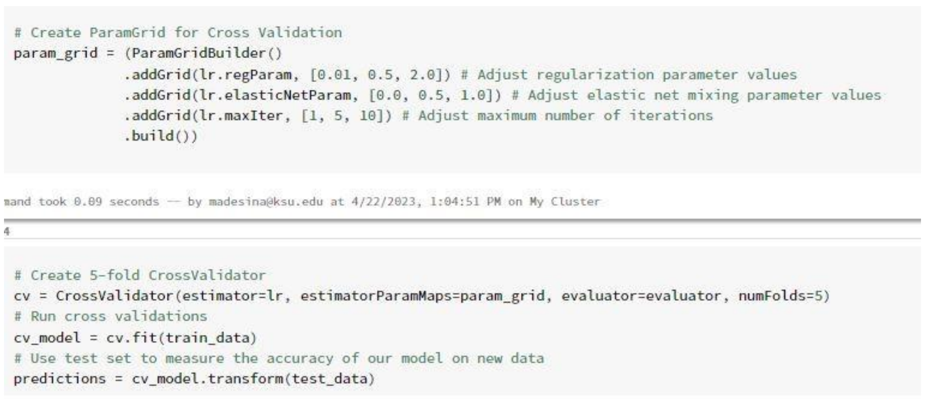
The basic idea of cross-validation is to repeatedly train the model on a subset of the data (training set) and evaluate its performance on a different subset (validation set). This process is repeated multiple times with different subsets used for training and validation in each iteration. The most commonly used type of cross-validation is "k-fold cross-validation", where the data is divided into k equally sized folds, and the model is trained k times, each time using k-1 folds for training and 1 fold for validation. The performance metrics, such as accuracy, precision, recall, etc., are calculated for each fold, and then averaged to obtain an overall estimate of the model's performance.

4.1.1. Cross Validation for Logistic Regression (5 Fold)

Creation of a **parameter grid** for cross-validation in a logistic regression model was done. The parameter grid is created using the ParamGridBuilder class, which allows you to specify different values for the hyperparameters of the logistic regression model. These hyperparameter values will be used in the cross-validation process to train and evaluate multiple logistic regression models with different hyperparameter combinations, and the best-performing model based on a chosen evaluation metric will be selected for further analysis or deployment.

Creation of 5-fold CrossValidator was done and the model was run using 5 folds, following by testing and measurement of accuracy on the testing dataset. The result will be discussed later in this chapter.

knitr::include_graphics("cvlrcrea.JPG")



```
# Create ParamGrid for Cross Validation
param_grid = (ParamGridBuilder()
             .addGrid(lr.regParam, [0.01, 0.5, 2.0]) # Adjust regularization parameter values
             .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0]) # Adjust elastic net mixing parameter values
             .addGrid(lr.maxIter, [1, 5, 10]) # Adjust maximum number of iterations
             .build())
```

```
mand took 0.09 seconds -- by madesina@ksu.edu at 4/22/2023, 1:04:51 PM on My Cluster
```

```
4
```

```
# Create 5-fold CrossValidator
cv = CrossValidator(estimator=lr, estimatorParamMaps=param_grid, evaluator=evaluator, numFolds=5)
# Run cross validations
cv_model = cv.fit(train_data)
# Use test set to measure the accuracy of our model on new data
predictions = cv_model.transform(test_data)
```

Figure 11 5-Fold Cross Validation Model Builder for Logistic Regression

Top Features Top features in Logistic regression Cross-Validation

knitr::include_graphics("logcr.JPG")

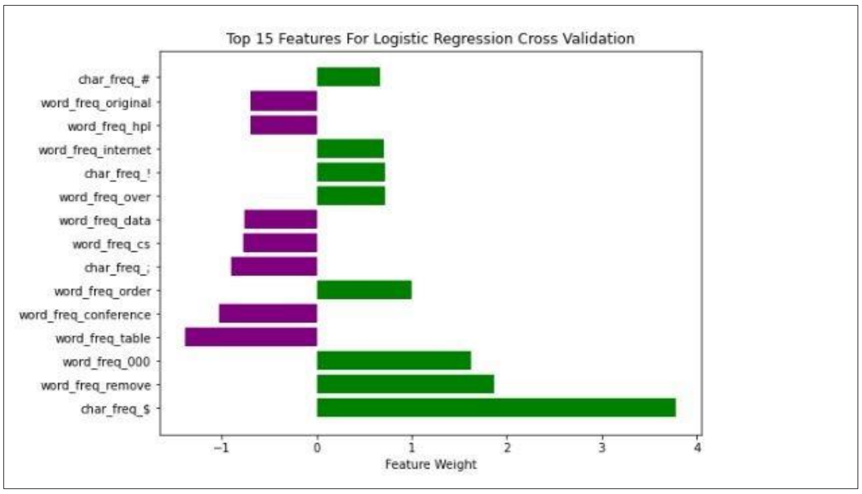


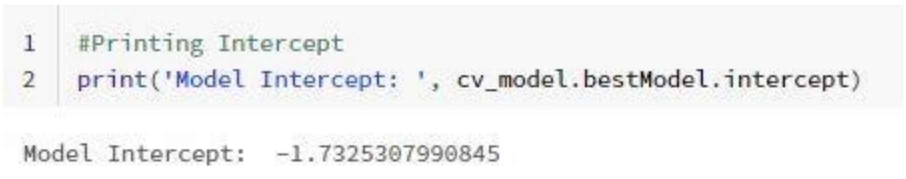
Figure 12 Top 15 Cross-Validation Model Feature Importance for Logistic Regression

-* Char_freq_$ is the top feature in this category, followed by word_freq_remove and word_freq_000.*

Intercept - Reported Model Intercept: -1.7325307990845. The negative value suggests that the model predicts a lower probability of spam (or a negative outcome) when all input features are set to zero.

4.2. Definition and analysis

knitr::include_graphics("cvlrint.JPG")



```
#Printing Intercept
print('Model Intercept: ', cv_model.bestModel.intercept)
```

```
Model Intercept:  -1.7325307990845
```

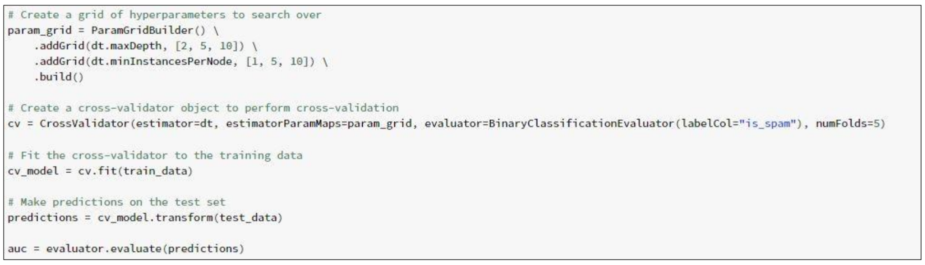
• This intercept is a little bit low in comparison to Logistic regression alone.

4.2.1. Cross Validation for Decision Tree (5 Fold)

Creation of a **parameter grid** for cross-validation in a decision tree model was done. The parameter grid is created using the ParamGridBuilder class, which allows you to specify different values for the hyperparameters of the decision tree model. These hyperparameter values will be used in the crossvalidation process to train and evaluate multiple decision tree models with different hyperparameter combinations, and the best-performing model based on a chosen evaluation metric will be selected for further analysis or deployment.

Creation of 5-fold CrossValidator was done and the model was run using 5 folds, following by testing and measurement of accuracy on the testing dataset. The result will be discussed later in this chapter.

knitr::include_graphics("cvdtcrea.JPG")



```
# Create a grid of hyperparameters to search over
param_grid = ParamGridBuilder() \
    .addGrid(dt.maxDepth, [2, 5, 10]) \
    .addGrid(dt.minInstancesPerNode, [1, 5, 10]) \
    .build()

# Create a cross-validator object to perform cross-validation
cv = CrossValidator(estimator=dt, estimatorParamMaps=param_grid, evaluator=BinaryClassificationEvaluator(labelCol="is_spam"), numFolds=5)

# Fit the cross-validator to the training data
cv_model = cv.fit(train_data)

# Make predictions on the test set
predictions = cv_model.transform(test_data)

auc = evaluator.evaluate(predictions)
```

Figure 13 5-Fold Cross Validation Model Builder for Decision Tree

Top Features Top features in Decision Tree Cross-Validation

knitr::include_graphics("decr.JPG")

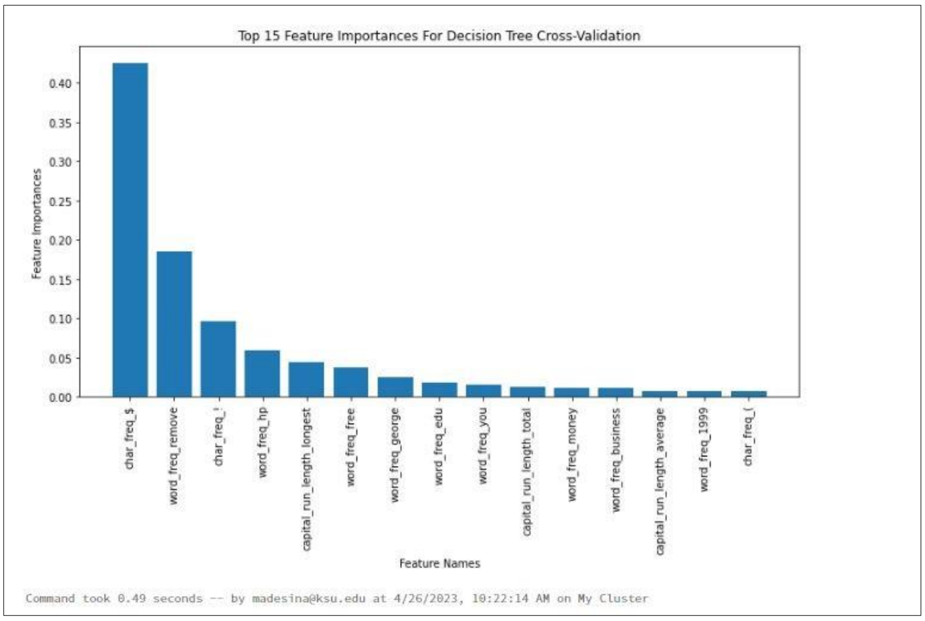


Figure 14 Top 15 Cross-Validation Model Feature Importance for Decision Tree

-* Char_freq_$ is the top feature in this category, followed by word_freq_remove and word_freq_!.*

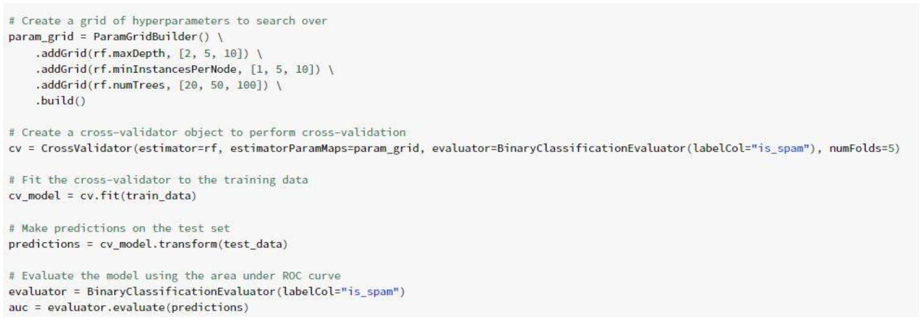
4.2.2. Cross Validation for Random Forest (5 Fold)

Creation of a **parameter grid** for cross-validation in a Random Forest model was done. The parameter grid is created using the ParamGridBuilder class, which allows you to specify different values for the hyperparameters of the Random Forest model. These hyperparameter values will be used in the cross-validation process to train and evaluate multiple decision tree models with different hyperparameter combinations, and the best-performing model based on a chosen evaluation metric will be selected for further analysis or deployment.

Creation of 5-fold CrossValidator was done and the model was run using 5 folds, following by testing and measurement of accuracy on the testing dataset. The result will be discussed later in this chapter.

knitr::include_graphics("cvrmcrea.JPG")

4.3. Definition and analysis

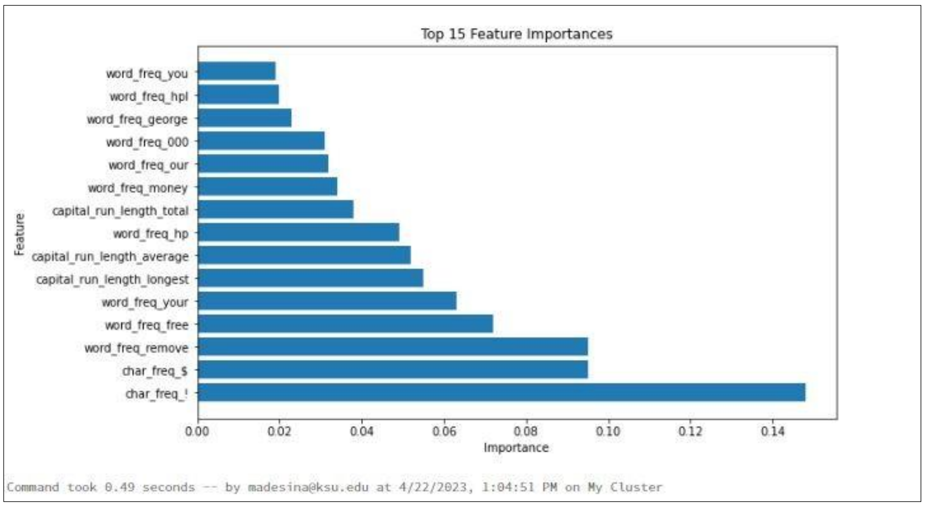


```
# Create a grid of hyperparameters to search over
param_grid = ParamGridBuilder() \
    .addGrid(rf.maxDepth, [2, 5, 10]) \
    .addGrid(rf.minInstancesPerNode, [1, 5, 10]) \
    .addGrid(rf.numTrees, [20, 50, 100]) \
    .build()

# Create a cross-validator object to perform cross-validation
cv = CrossValidator(estimator=rf, estimatorParamMaps=param_grid, evaluator=BinaryClassificationEvaluator(labelCol="is_spam"), numFolds=5)

# Fit the cross-validator to the training data
cv_model = cv.fit(train_data)

# Make predictions on the test set
predictions = cv_model.transform(test_data)

# Evaluate the model using the area under ROC curve
evaluator = BinaryClassificationEvaluator(labelCol="is_spam")
auc = evaluator.evaluate(predictions)
```

Figure 15 5-Fold Cross Validation Model Builder for Random Forest

Top Features Top features in Random Forest Cross-Validation

knitr::include_graphics("featgraph.JPG")



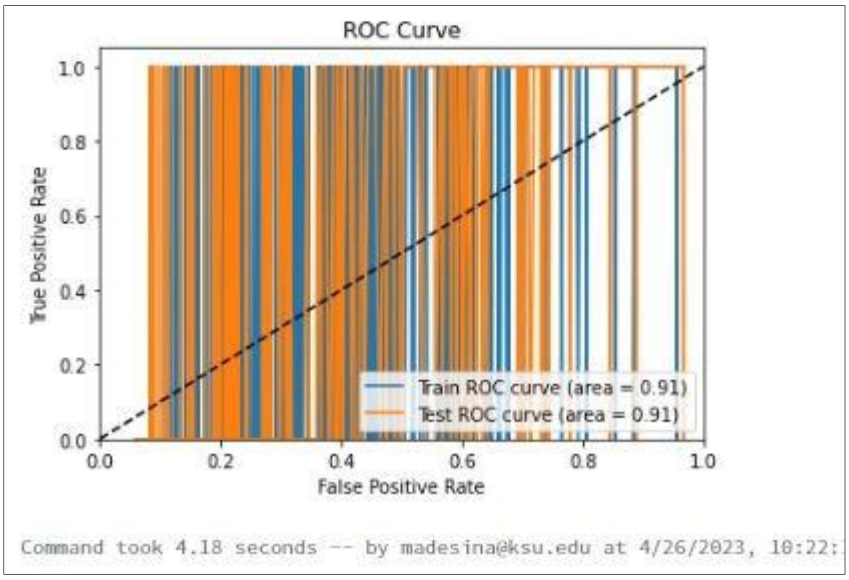
```
Command took 0.49 seconds -- by madesina@ksu.edu at 4/22/2023, 1:04:51 PM on My Cluster
```

Figure 16 Top 15 Cross-Validation Model Feature Importance for Random Forest

-* Char_freq_! is the top feature in this category, followed by word_freq_remove and word_freq_your.*

Graph Showing Area Under ROC

knitr::include_graphics("auc.JPG")



```
Command took 4.18 seconds -- by madesina@ksu.edu at 4/26/2023, 10:22:
```

Figure 17 Random Forest ROC Curve

5. Discussion of results

**Here is a breakdown of scores for each model:

Logistic Regression Cross Validation:

- Area Under ROC: 0.967
- Precision: 0.915
- Recall: 0.915
- F1 Score: 0.915
- Accuracy: 0.915

Decision Tree Cross Validation:

- Area Under ROC: 0.827
- Precision: 0.924
- Recall: 0.923
- F1 Score: 0.923
- Accuracy: 0.923

Random Forest Cross Validation:

- Area Under ROC: 0.982
- Precision: 0.942
- Recall: 0.941
- F1 Score: 0.941
- Accuracy: 0.941

Now let's compare the performance of these models:

Area Under ROC (Receiver Operating Characteristic) measures the ability of a model to correctly classify between positive and negative instances. A higher value indicates better performance, and in this case, the Random Forest model has the highest value of 0.982, followed by Logistic Regression with 0.967, and Decision Tree with 0.827.

Precision measures the accuracy of positive predictions made by the model. A higher precision value indicates fewer false positives. In this case, Decision Tree has the highest precision of 0.924, followed by Random Forest with 0.942, and Logistic Regression with 0.915.

Recall measures the ability of the model to identify all the positive instances correctly. A higher recall value indicates fewer false negatives. In this case, Logistic Regression and Decision Tree have the same recall value of 0.915, followed by Random Forest with 0.941.

F1 Score is the harmonic mean of precision and recall, and it provides a balanced measure of model performance. In this case, all three models have similar F1 Scores, with Decision Tree, Random Forest, and Logistic Regression having values of 0.923, 0.941, and 0.915 respectively.

Accuracy measures the overall correctness of the model's predictions. In this case, all three models have similar accuracy values, with Decision Tree, Random Forest, and Logistic Regression having values of 0.923, 0.941, and 0.915 respectively.

Based on the performance metrics, the **Random Forest model** has the highest Area Under ROC, precision, recall, and F1 Score among the three models. Therefore, the Random Forest model appears to be the best model out of the three for this specific dataset and problem, based on the provided performance metrics.

6. Cross-validation

6.1. Features

6.1.1. Definition

Feature refers to an individual variable or predictor that is used as input to a model for making predictions or classifications. Features are also known as "input variables," "explanatory variables," or "independent variables" in different fields of study.

In a machine learning model, features are used to represent the characteristics or attributes of the data that the model uses to learn patterns and make predictions. For example, in a spam detection model, features could include the frequency of certain words, the presence of certain characters or symbols, or other relevant information extracted from the emails being analyzed. These features are used as input to the model during the training phase, where the model learns the relationships between the features and the target variable (e.g., spam or nonspam) from the labeled data.

Features play a critical role in the performance and interpretability of a model. The selection and engineering of appropriate features can greatly impact the accuracy, interpretability, and generalization ability of a model. It is important to carefully choose relevant features that capture the relevant information in the data and remove any irrelevant or redundant features that may introduce noise or lead to overfitting. Feature engineering, which involves selecting, transforming, and creating new features from the original data, is often an important step in the machine learning pipeline to improve the performance of a model.

feature weight refers to the importance or contribution of each individual feature or predictor in influencing the predictions or classifications made by the model. Feature weights are also known as “coefficients,” “parameters,” or “weights” in different machine learning algorithms.

During the training process of a machine learning model, the model learns the optimal values for the feature weights that best fit the data and minimize the prediction errors. These feature weights determine the relative importance of each feature in the model’s decision-making process. In some models, such as linear regression or logistic regression, feature weights are directly estimated and used to compute predictions or probabilities. In other models, such as decision trees or neural networks, feature weights may not be explicitly available, but their impact on the model’s predictions can still be understood by analyzing the structure or activations of the model.

Feature weights are crucial for interpreting the model’s behavior and understanding the importance of each feature in the model’s predictions. **Positive feature weights** indicate that an increase in the value of the corresponding feature positively contributes to the predicted outcome, while **negative feature weights** indicate the opposite. The magnitude of the feature weights represents the relative importance of each feature, with larger absolute values indicating stronger influence on the model’s predictions.

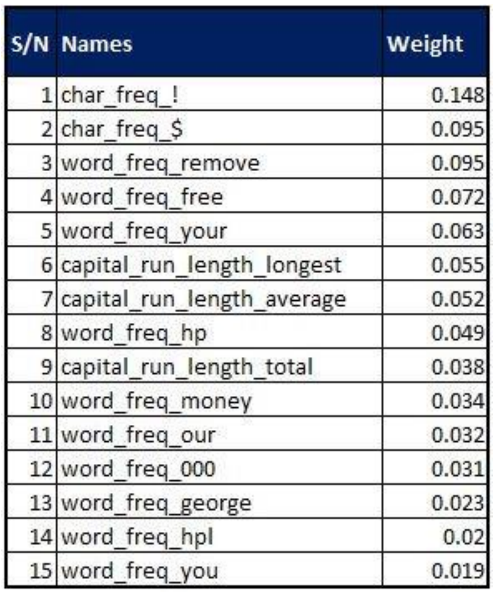
6.2. Identification of most significant features.

Feature Importance was developed for all the three ML classification models, Please refer to the pyspark code section to see the most important features in each model. However, Based on the best model selected which is **Random Forest Cross Validation Model**, The following are the most important features in the dataset.

knitr::include_graphics("feat.JPG")

6.3. Identification of most significant features.

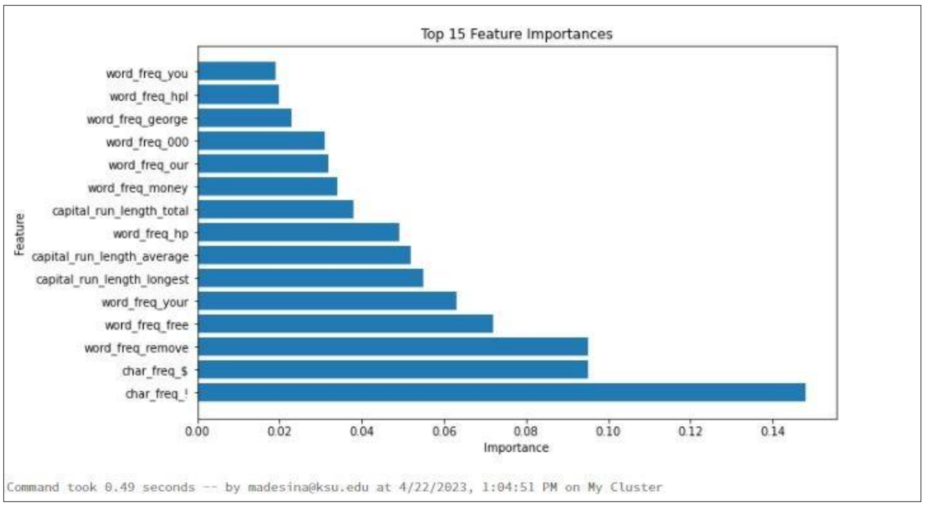
Table 1 Most Significant Features



S/N	Names	Weight
1	char_freq_!	0.148
2	char_freq_$	0.095
3	word_freq_remove	0.095
4	word_freq_free	0.072
5	word_freq_your	0.063
6	capital_run_length_longest	0.055
7	capital_run_length_average	0.052
8	word_freq_hp	0.049
9	capital_run_length_total	0.038
10	word_freq_money	0.034
11	word_freq_our	0.032
12	word_freq_000	0.031
13	word_freq_george	0.023
14	word_freq_hpl	0.02
15	word_freq_you	0.019

• char_freq_! is the most important feature based on Random Forest algorithm selected, a graphical representation of this and other top 14 important features and their weight is shown below:

knitr::include_graphics("featgraph.JPG")



```
Command took 0.49 seconds -- by madesina@ksu.edu at 4/22/2023, 1:04:51 PM on My Cluster
```

Figure 18 Top 15 Cross-Validation Model Feature Importance for Random Forest

6.3.1. Discussing the Important Features

char_freq_!: The feature "char_freq_!" has an importance value of 0.148, which indicates that it is considered one of the most important features for predicting spam or non-spam emails by the random forest model. This feature represents the frequency of the exclamation mark character "!" in the email. A higher value of this feature suggests that the email contains more exclamation marks, which may be indicative of spammy or promotional content.

char_freq_$: The feature has an importance value of 0.095, making it another important feature in the random forest model. This feature represents the frequency of the dollar sign character in the email. A higher value of this feature suggests that the email contains more occurrences of dollar signs, which could be indicative of financial-related spam or phishing attempts.

word_freq_remove: The feature "word_freq_remove" has an importance value of 0.095, indicating its importance in the random forest model. This feature represents the frequency of the word "remove" in the email.

A higher value of this feature suggests that the email contains more occurrences of the word "remove," which could be indicative of spammy content related to removing or eliminating something.

word_freq_free: The feature "word_freq_free" has an importance value of 0.072, suggesting its relevance in predicting spam or non-spam emails. This feature represents the frequency of the word "free" in the email. A higher value of this feature suggests that the email contains more occurrences of the word "free," which could be indicative of promotional or spammy content related to free offers or giveaways.

word_freq_your: The feature "word_freq_your" has an importance value of 0.063, indicating its importance in the random forest model. This feature represents the frequency of the word "your" in the email. A higher value of this feature suggests that the email contains more occurrences of the word "your," which could be indicative of personalized or targeted content, including both spam and legitimate emails.

capital_run_length_longest: This feature measures the length of the longest run of capital letters in the email text. The longer the run of capital letters, the more likely the email is to be spam. This is because spammers often use capital letters in an attempt to catch the recipient's attention.

capital_run_length_average: The feature "capital_run_length_average" has an importance value of 0.052, indicating its relevance in predicting spam or non-spam emails. This feature represents the average length of consecutive sequences of capital letters in the email. A higher value of this feature suggests that the email contains longer average sequences of capital letters, which could be indicative of spammy or promotional

6.4. Identification of most significant features

word_freq_hp: The feature "word_freq_hp" has an importance value of 0.049, suggesting its relevance in the random forest model. This feature represents the frequency of the word "hp" in the email. A higher value of this feature suggests that the email contains more occurrences of the word “hp,” which could be indicative of spammy or promotional content related to the brand “HP” or similar products.

capital_run_length_total: The feature “capital_run_length_total” has an importance value of 0.038, indicating its importance in the random forest model. This feature measures the total length of runs of capital letters in the email text. Similar to the previous two features, the higher the total length of capital letter runs, the more likely the email is to be spam.

word_freq_money: This feature measures the frequency of the word “money” in the email text. The higher the frequency, the more likely the email is to be spam. This could be because spammers often try to entice recipients to invest or spend money.

word_freq_our: This feature measures the frequency of the word “our” in the email text. A higher frequency of the word “our” may indicate that the email is more likely to be spam, as spammers often use possessive pronouns like “our” to create a sense of familiarity and gain the recipient’s trust.

word_freq_000: This feature measures the frequency of the sequence “000” in the email text. The higher the frequency, the more likely the email is to be spam. This could be because spammers often include sequences of numbers in an attempt to bypass spam filters.

word_freq_george: This feature measures the frequency of the name “George” in the email text. The presence of this name may indicate that the email is more likely to be spam, as spammers often use generic names like “George” to create fake sender identities.

word_freq_hpl: This feature measures the frequency of the acronym “HPL” in the email text. The lower the frequency, the less likely the email is to be spam. This could be because “HPL” may be associated with legitimate content or sources, and thus its presence in an email may indicate that the email is not spam.

word_freq_you: This feature measures the frequency of the word “you” in the email text. The lower the frequency, the less likely the email is to be spam. This could be because legitimate emails often address the recipient directly using “you,” while spammers may use different forms of impersonal language.

6.5. How to Take Advantage of the Findings.

The importance scores of features obtained from the random forest model trained on the spambase.data can be used in several ways to improve the performance of the model or gain insights into the problem of spam classification. Here are some potential ways to take advantage of this finding:

Feature selection: Based on the importance scores, one can select a subset of the most important features to train a model, which can lead to better performance or faster training times. In this case, we can select the top few features with highest importance scores such as “char_freq_!”, “char_freq_$”, “word_freq_remove”, “word_freq_free”, and “word_freq_your” and discard the less important features.

Feature engineering: The importance scores can guide the process of feature engineering, where new features can be created based on the existing ones to improve the model’s performance. For example, we can create a new feature that combines the frequencies of the exclamation mark and dollar sign characters, given their high importance scores, or engineer new features based on domain knowledge or data analysis.

Model interpretation: The importance scores can provide insights into the problem of spam classification, by revealing which features are most relevant for distinguishing spam from non-spam messages. For example, the high importance of the “word_freq_remove” feature suggests that the presence of words like “remove” may be a strong indicator of spam, while the importance of “word_freq_your” suggests that the use of personalized language may be more common in non-spam messages.

Model comparison: The importance scores can be used to compare the performance of different models or algorithms on the same dataset, and identify which features are consistently important across models. For example, we can compare the importance scores of random forests with those of logistic regression or decision trees, and see which features are consistently ranked high, which can help us choose the most appropriate model for the problem.

Monitoring and maintenance: The feature importance values can also be used for ongoing monitoring and maintenance of the model in a production environment. By regularly monitoring the importance values, you can detect any changes in the importance of features over time and take necessary actions, such as updating the model or reevaluating the relevance of certain features in the current context.

7. Discussing the Best ML Algorithm ML algorithm from a theoretical point of view

Based on the given performance metrics, the "Random Forest" algorithm is the best performing algorithm among the three (Logistic Regression, Decision Tree, and Random Forest) in terms of Area Under the Receiver Operating Characteristic (ROC) curve, Precision, Recall, F1 Score, and Accuracy. Here's a theoretical discussion on why the Random Forest algorithm may be superior to the other two:

Ensemble Technique: Random Forest is an ensemble technique that combines multiple decision tree models to make predictions. It creates a forest of decision trees, where each tree is trained on a random subset of features and data samples. This allows for better generalization and reduction of overfitting compared to single decision trees used in Decision Tree and Logistic Regression.

Robust to Overfitting: Random Forests are less prone to overfitting compared to single decision trees. The random sampling of features and data samples during tree construction, and the averaging of predictions from multiple trees, helps in reducing variance and improving model stability. This can result in better performance on unseen data, as reflected in the higher Area Under ROC and accuracy scores.

High Predictive Accuracy: The given performance metrics, such as

Precision, Recall, and F1 Score, are also higher for Random Forest com-CHAPTER 5. DISCUSSING THE BEST ML ALGORITHM ML ALGORITHM FROM A THEORET

pared to Decision Tree and Logistic Regression. This indicates that Random Forest is able to achieve better predictive accuracy in terms of both positive and negative predictions, resulting in a balanced performance.

Ability to Capture Non-Linear Relationships: Random Forest can capture complex non-linear relationships in the data, as it can build deep and wide decision trees with multiple splits. In contrast, Logistic Regression and Decision Tree are generally limited in their ability to capture non-linear patterns in the data, which can impact their performance on datasets with non-linear relationships.

Handling of Missing Values: Random Forest can handle missing values in the data effectively, as it can make predictions based on the available features without imputing missing values. In contrast, Logistic Regression and Decision Tree may require additional handling of missing values, which can introduce biases or reduce model performance.

Feature Importance: Random Forest can provide feature importance measures, which can help in identifying the most important features for making accurate predictions. This can aid in feature selection and feature engineering efforts, and provide better insights into the underlying data patterns.

8. Conclusion

In summary, Random Forest, as an ensemble technique, has several advantages over single algorithms like Logistic Regression and Decision Tree, including better generalization, robustness to overfitting, ability to capture non-linear relationships, handling of missing values, and feature importance measures. These theoretical advantages may explain the superior performance of Random Forest in the given performance metrics compared to the other two algorithms. However, it's important to note that the best algorithm ultimately depends on the specific characteristics of the dataset, problem domain, and modeling goals, and should be chosen based on thorough experimentation and evaluation.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Mitchell, T. M. (1997). Machine Learning. McGraw-Hill Education.

[2] UCI Machine Learning Repository. (2024). Spambase Data Set. University of California, Irvine.

[3] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An Introduction to Statistical Learning: with Applications in R. Springer.

[4] Quinlan, J. R. (1986). Induction of Decision Trees. Machine Learning, 1(1), 81-106.

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

[6] Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. International Joint Conference on Artificial Intelligence (IJCAI).

[7] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437.

[8] Zhang, M. L., & Zhou, Z. H. (2014). A review on multi-label learning algorithms. IEEE Transactions on Knowledge and Data Engineering, 26(8), 1819-1837.

[9] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, 3, 1157-1182.

[10] Domingos, P. (2012). A few useful things to know about machine learning. Communications of the ACM, 55(10), 78-87.