Analysis of the SMS Spam Collection Dataset

Eleni Baltzi

October 28, 2025

A. Introduction

The instructions provided for the LDA and QDA and SMOTE analysis of the iris dataset [1] were adapted in order to perform similar analysis for the SMS Spam Collection dataset.

B. LDA Analysis

Loading Required Packages

To perform data processing and analysis, the following R packages are used (Listing 1) [2], [7], [8]:

```r
packages_names <- c("tidyverse",  # for data wrangling
"MASS",      # for LDA
"caret",     # for data splitting and confusion matrix
"smotefamily"    # for SMOTE oversampling (if using synthetic augmentation)
)

invisible(lapply(packages_names, function(x) {
if (!require(x, character.only = TRUE)) {
install.packages(x, dependencies = TRUE)
library(x, character.only = TRUE)
}
}))
```

Listing 1: Code block for loading the initial required packages

First was defined a list of required packages and then checks if each package is installed (Listing 2). If a package is missing, it installs it before loading it into the R session.

Loading the Dataset

The dataset is read from a CSV file [3] and stored in a dataframe. The column names are also defined explicitly:

```r
#Load the dataset
sms_data <- read.csv("/home/fenrir/T l chargements/sms+spam+collection/SMSSpamCollection.csv", stringsAsFactors = FALSE)
colnames(sms_data) <- c("label", "text")

#Verify unique labels
unique_labels <- unique(sms_data$label)
print(unique_labels)  # Should show only 'ham' and 'spam'
```

Listing 2: Code block for loading the dataset and verifing the unique labels present in the dataset

The dataset consists of two columns:

- **label**: This column indicates whether an SMS message is spam or not. The values are either `ham` (not spam) or `spam` (spam).
- **text**: This column contains the content of the SMS message.

The output of the script confirms that the dataset consists of only two unique labels: `"ham"` and `"spam"`. This means the dataset does not contain any additional categories, making it a binary classification problem.

```r
#Set labels as factors
sms_data$label <- factor(sms_data$label)

#Check factor levels
print(levels(sms_data$label))
```

Listing 3: Converting labels to factors and verifying them.

This step (Listing 3) ensures that the labels are treated as categorical variables, which is necessary for classification tasks. The output confirms that the factor levels remain `"ham"` and `"spam"`, indicating that the conversion was successful.

Creating a Corpus and Document-Term Matrix

To preprocess the text data, a corpus is created, and common preprocessing steps such as converting to lowercase, removing punctuation, numbers, and stopwords, as well as stripping whitespace, are applied (Listing 4):

```r
library(tm)
#Create a corpus from the text data
corpus <- Corpus(VectorSource(sms_data$text))

#Preprocess the corpus - converting to lowercase, removing punctuation,
    numbers, stopwords, and whitespace.

corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, stripWhitespace)

#Create a Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)

#Check the DTM structure
print(dtm)
```

Listing 4: Creating a corpus from the text data

The Document-Term Matrix (DTM) is used to represent the frequency of terms in documents (Listing 4). However, in the given output:

```
<<DocumentTermMatrix (documents:
0, terms: 0)>>
Non-/sparse entries: 0/0
Sparsity: 100Maximal term
length: 0
Weighting: term frequency
(tf)
```

this suggests that the document-term

matrix is empty. Possible reasons for this issue include:

- The column containing the message text was misnamed in the dataframe (should be `text` instead of `Message`).
- The dataset might be empty or improperly loaded.
- Preprocessing steps may have removed all words.

Verifying the dataset and checking the column names should help resolve this issue (Listing 5).

```r
#Find empty documents
empty_docs <- which(rowSums(as.matrix(dtm)) == 0)

#Print out the IDs of empty documents for review
cat("Empty document indices:", empty_docs, "\n")

#Filter out the empty documents from the corpus
corpus <- corpus[-empty_docs]

#Recreate the DTM without the empty documents
dtm <- DocumentTermMatrix(corpus)
```

Listing 5: Finding the empty documents

The Document-Term Matrix (DTM) is used to represent the frequency of terms in documents. However, in the given output:

```
Empty document indices:
```

This suggests that some documents contained no meaningful words after preprocessing, resulting in an empty DTM. To resolve this, the script identifies and removes such empty documents before recreating the DTM (Listing 5).

Preparing Text Data for Modeling

Once the document-term matrix is cleaned, it is converted into a data frame for further analysis(Listing 6):

```r
library(tm)

#Check variable names
colnames(train_data) <- make.names(colnames(train_data), unique = TRUE)

corpus <- Corpus(VectorSource(sms_data$text))
dtm <- DocumentTermMatrix(corpus, control = list(removePunctuation =
    TRUE, stopwords = TRUE))

#Convert DTM to data frame
sms_matrix <- as.matrix(dtm)
train_data <- as.data.frame(sms_matrix)
train_data$label <- sms_data$label

#Check structure
str(train_data)
```

Listing 6: Preparing Text Data for Modeling

The dataset now consists of numerical values representing the frequency of words in each SMS message. The structure of traindata reveals that it contains 5533 observations (SMS messages) and 7666 variables (unique terms). This transformation is crucial for applying machine learning models to classify SMS messages as spam or ham.

Training the LDA Model

Linear Discriminant Analysis (LDA) is applied to classify messages (Listing 7):

```
lda_data <- as.data.frame(sampled_train_dtm)
# Add labels to the dataset
lda_data$label <- sampled_train_labels

lda_model <- lda(label ~ ., data = lda_data)
print(lda_model)
```

Listing 7: Linear Discriminant Analysis (LDA)

The LDA model learns patterns distinguishing spam from ham messages based on word frequencies.

Parallel Processing for Computation Efficiency

To speed up computation, parallel processing [11] is utilized (Listing 8):

```
# Use one less than the total number of cores available
registerDoParallel(detectCores() - 1)
```

Listing 8: Parallel Processing for Computation Efficiency

This approach leverages multiple CPU cores to accelerate the LDA training process.

LDA Analysis Output

The output of the LDA model [2] [5] includes:

- **Prior probabilities**: The proportion of messages belonging to each category (ham vs. spam).
- **Group means**: The average frequency of terms in each class.
- **Linear discriminant coefficients**: The importance of each term in distinguishing spam from ham.

Sample output:

Prior probabilities of groups: ham spam 0.865 0.135

Coefficients of linear discriminants: free 1.66776769 text 3.59031692 win 5.58536025 claim 8.94428555 mobile 3.92086701 stop 5.37929770 urgent 2.57729192

This output indicates that words like "claim," "win," "urgent," and "stop" strongly influence the classification of a message as spam.

Confusion Matrix Calculation

The following R script computes the confusion matrix for the LDA model (Listing 9):

```r
#Convert to a data frame before making predictions
predictions <- predict(lda_model, as.data.frame(sampled_train_dtm))$
    class

#Compute confusion matrix
conf_matrix <- confusionMatrix(predictions, sampled_train_labels) print
    (conf_matrix)
```

Listing 9: Confusion Matrix Calculation

Confusion Matrix Output

The output of the confusion matrix is as follows:

Confusion Matrix and Statistics

```
Reference Prediction ham spam ham
851 50 spam 14 85
           Accuracy : 0.936
        95% CI : (0.919, 0.9504)
No Information Rate : 0.865
P-Value [Acc > NIR] : 4.327e-13

              Kappa : 0.6912


Mcnemar's Test P-Value : 1.214e-05
        Sensitivity : 0.9838
        Specificity : 0.6296
     Pos Pred Value : 0.9445
     Neg Pred Value : 0.8586
         Prevalence : 0.8650
     Detection Rate : 0.8510

Detection Prevalence : 0.9010
Balanced Accuracy : 0.8067
   'Positive' Class : ham
```

- **Accuracy:** The model achieves an accuracy of 93.6%, meaning that it correctly classifies 93.6% of SMS messages.
- **Sensitivity (Recall):** 98.38% of actual ham messages are correctly classified as ham.
- **Specificity:** 62.96% of actual spam messages are correctly classified as spam, indicating that the model has some difficulty distinguishing spam.
- **Positive Predictive Value (Precision):** When the model predicts "ham," it is correct 94.45% of the time.
- **Negative Predictive Value:** When the model predicts "spam," it is correct 85.86% of the time.
- **Kappa Score:** 0.6912, suggesting a moderate agreement between the predicted and actual classifications.
- **McNemar's Test P-Value:** A significant p-value (1.214e-05) indicates that the misclassification rates for ham and spam are not equal, highlighting an imbalance in the classification performance.

C. QDA Analysis

Initial Data and Package handling

To perform the QDA analysis [2], [6] the necessary packages and the dataset were loaded (Listing 10). Extra spaces were removed and labels were converted into factors. Finally, NA values were removed. A similar data clearing and handling was performed just like in section B. A corpus was created (Listing 11) the data was split (Listing 12), a document-text matrix was created (Listing 12) and then it was converted to a dataframe (Listing 13) from which the NA columns were removed (Listing 14).

```r
library(MASS)
library(tm)
library(tidytext)
# Load the dataset
file_path <- "/home/fenrir/Downloads/LDA-QDA/SMSSpamCollection.csv"
df <- read.csv(file_path, header = FALSE, sep = ",", stringsAsFactors =
    FALSE)

# Rename columns
colnames(df) <- c("Label", "Message")

# Remove extra spaces
df$Label <- trimws(df$Label)
df$Message <- trimws(df$Message)

# Convert labels to factors
df$Label <- factor(df$Label, levels = c("ham", "spam"))

# Remove NA values
df <- na.omit(df)
```

Listing 10: Preliminary data and packages handling

```r
library(tm)

# Create a corpus from the text data
corpus <- Corpus(VectorSource(sms_data$Message))

# Preprocess the corpus - converting to lowercase, removing
    punctuations, stopwords, and stemming might be done here.
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removeWords, stopwords("en"))
corpus <- tm_map(corpus, stripWhitespace)


# Create a Document-Term Matrix
dtm <- DocumentTermMatrix(corpus)

# Check the DTM structure
print(dtm)
```

Listing 11: Creating the Corpus

```r
library(tm)
library(tidytext)
# Split data into training and testing sets (80%-20%)
set.seed(42)
trainIndex <- createDataPartition(df$Label, p = 0.8, list = FALSE)
trainData <- df[trainIndex, ]
testData  <- df[-trainIndex, ]
# Create a document-term matrix
train_corpus <- Corpus(VectorSource(trainData$Message))
corpus <- Corpus(VectorSource(sms_data$text))
test_corpus <- Corpus(VectorSource(testData$Message))

train_dtm <- DocumentTermMatrix(train_corpus, control = list(weighting
    = weightTfIdf))
test_dtm <- DocumentTermMatrix(test_corpus, control = list(weighting =
    weightTfIdf))
```

Listing 12: Data Split and Creating a document-term matrix

```r
# Convert to dataframe
train_tfidf <- as.data.frame(as.matrix(train_dtm))
test_tfidf <- as.data.frame(as.matrix(test_dtm))

train_tfidf <- as.data.frame(lapply(train_tfidf, as.numeric))
test_tfidf <- as.data.frame(lapply(test_tfidf, as.numeric))
train_tfidf <- na.omit(train_tfidf)
test_tfidf <- na.omit(test_tfidf)

str(train_tfidf)
```

Listing 13: Convert to dataframe

```r
# Remove columns that are entirely NA
train_tfidf <- train_tfidf[, colSums(is.na(train_tfidf)) == 0, drop =
    FALSE]
test_tfidf <- test_tfidf[, colSums(is.na(test_tfidf)) == 0, drop =
    FALSE]
dim(train_tfidf)
```

Listing 14: Removing NA columns

TF-IDF Transformation and PCA for Dimensionality Reduction

The text data were processed by applying Term Frequency-Inverse Document Frequency (TF-IDF) [9] weighting, reducing dimensionality using Principal Component Analysis (PCA) [10], and preparing the dataset for classification (Listing 15).

TF-IDF Recalculation

The Document-Term Matrix (DTM) is created for both training and testing datasets, with TF-IDF weighting applied to highlight important words. The parameter `bounds = list(global = c(2, Inf))` ensures that only words appearing in at least two documents are retained, reducing noise caused by rare words.

Aligning Training and Testing Data

To maintain consistency, only common words (features) between the training and testing datasets are retained. This prevents issues where the test set contains words that were not present in the training set.

Principal Component Analysis (PCA)

Dimensionality reduction is performed using PCA, selecting the first 50 principal components to preserve the most important variations. Reducing dimensionality helps prevent overfitting and improves computational efficiency while preserving critical information.

Restoring Class Labels

After PCA transformation, the original class labels (`ham`, `spam`) are reattached to the reduced datasets. This ensures that the transformed data can still be used for classification.

```r
# Recompute TF-IDF without removing too many words
train_dtm <- DocumentTermMatrix(train_corpus, control = list(weighting
   = weightTfIdf, bounds = list(global = c(2, Inf))))
test_dtm  <- DocumentTermMatrix(test_corpus, control = list(weighting =
     weightTfIdf, bounds = list(global = c(2, Inf))))

# Convert to dataframe again
common_cols <- intersect(colnames(train_tfidf), colnames(test_tfidf))
train_tfidf <- train_tfidf[, common_cols, drop = FALSE]
test_tfidf <- test_tfidf[, common_cols, drop = FALSE]


#PCA
pca_model <- prcomp(train_tfidf, center = TRUE, scale. = TRUE)
train_pca <- as.data.frame(predict(pca_model, train_tfidf)[, 1:50])
test_pca <- as.data.frame(predict(pca_model, test_tfidf)[, 1:50])
print(pca_model)


# Add labels back
train_pca$Label <- trainData$Label
test_pca$Label <- testData$Label
```

Listing 15: Recompute TF-IDF and handling hyperdimentionality

```r
# Train QDA model
qda_model <- qda(Label ~ ., data = train_pca)
print(qda_model)
```

Listing 16: Training the QDA model

Group Means for Principal Components

The group means indicate the average values of each **principal component (PC1 - PC50)** for both "ham" and "spam" classes.

- **PC1 & PC2**:
  - Ham: $PC1 = 0.177$, $PC2 = 0.153$
  - Spam: $PC1 = -1.152$, $PC2 = -0.991$

Spam emails have much lower values on the first two principal components than ham emails. This suggests that these components play a significant role in distinguishing spam from ham.

- **PC4 (Large Difference in Means)**:
  - Ham: -0.371
  - Spam: 2.407

PC4 is strongly positive for spam

but negative for ham. This suggests that **this feature strongly influences spam classification**.

- **PC6 & PC7**:
  - Ham: $PC6 = 0.154$, $PC7 = -0.136$
  - Spam: $PC6 = -0.999$, $PC7 = 0.881$

  Strong differences in these components indicate they also play a role in distinguishing spam.

- **PC10 and Beyond (Minimal Differences)**:
  - For most principal components beyond PC10, the differences between ham and spam are much smaller.

  This suggests that **the first few principal components contain most of the discriminatory power, while later PCs contribute less.**

Classification

- **Strong Separation in Early Principal Components**
  - The **first few PCs (PC1–PC7, especially PC4)** show **large differences in means** between spam and ham.
  - This suggests that these **early components capture most of the variation needed for classification**.

- **High Imbalance in Classes**
  - Since the dataset contains **86.65% ham messages**, the model may be **biased toward predicting ham**.
  - This could lead to **high false-negative rates**, where spam messages are misclassified as ham.

- **PCA Effectiveness**
  - The **first few PCs capture most of the variance** in distinguishing spam from ham.
  - The later PCs have much smaller differences, indicating they **do not contribute significantly to classification**.

```r
# Make predictions & Evaluate the model
library(tm)
library(MASS)
library(caret)
library(doParallel)
library(ggplot2)
# Make predictions
predictions <- predict(qda_model, test_pca)$class

# Evaluate the model
conf_matrix <- confusionMatrix(predictions, test_pca$Label)
print(conf_matrix)
```

Listing 17: Making predictions and evaluating the QDA model

Confusion Matrix Output

The output of the confusion matrix is as follows (Lisrting 17):

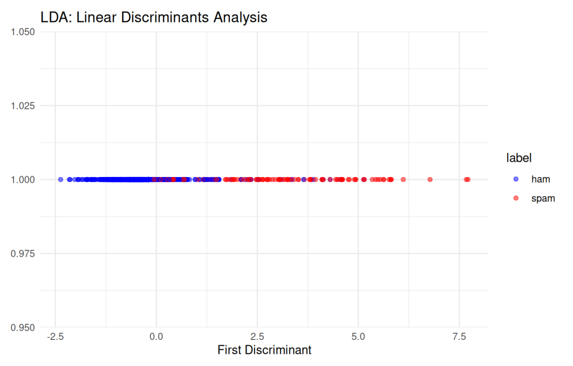
Confusion Matrix and Statistics

```
Confusion Matrix and Statistics
```

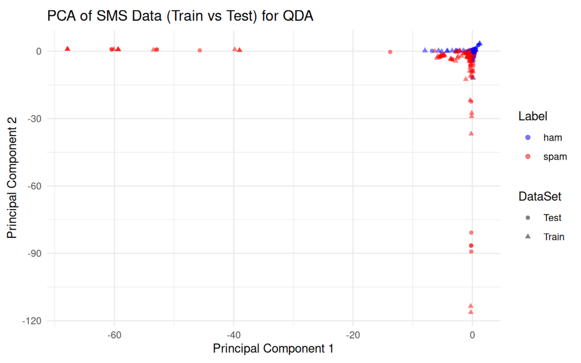
```
                                            Mcnemar's Test P-Value : 0.01156
          Reference
Prediction ham spam                            Sensitivity : 0.9353
      ham  897   36                            Specificity : 0.7551
      spam  62  111                         Pos Pred Value : 0.9614
                                            Neg Pred Value : 0.6416
               Accuracy : 0.9114               Prevalence : 0.8671
                 95% CI : (0.8931, 0.9275)  Detection Rate : 0.8110
    No Information Rate : 0.8671      Detection Prevalence : 0.8436
    P-Value [Acc > NIR] : 3.09e-06       Balanced Accuracy : 0.8452

                  Kappa : 0.6424          'Positive' Class : ham
```

D. Model Comparison

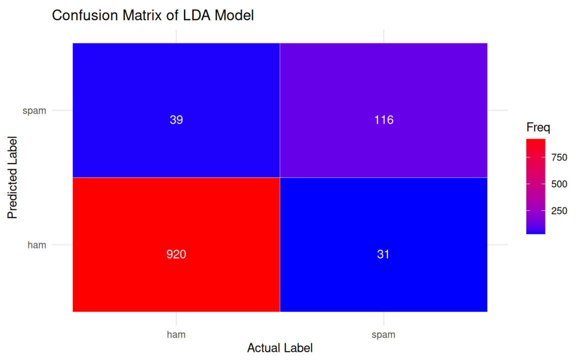


(a) LDA: Linear Discriminants Analysis

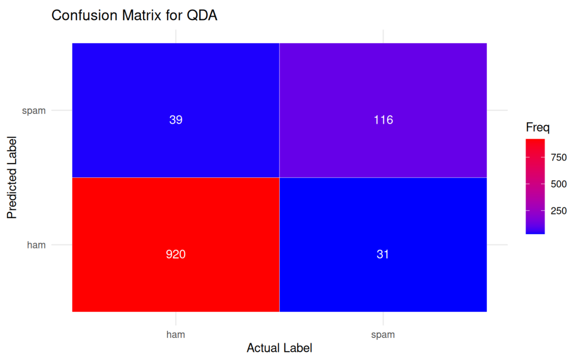


(b) PCA of SMS Data (Train vs Test) for QDA

Figure 1: Comparison of LDA and QDA



(a) Confusion Matrix of LDA Model



(b) Confusion Matrix for QDA

Figure 2: LDA-QDA Confusion Matrices

LDA-QDA Discriminants and PCA Plot

- **LDA Linear Discriminants Analysis**
  The plot (Figure 3a) represents the first *linear discriminant* of the LDA model, where data points are color-coded based on their labels (red for spam, blue for ham). The LDA projection shows that most data points are concentrated in the same region, indicating that LDA finds a single linear boundary to separate the classes. However, there is significant **overlap** between spam and ham messages, suggesting that a simple linear boundary might not be effective for classification.

- **PCA of SMS Data for QDA**
  This Principal Component Analysis (PCA) plot 6b) visualizes the dataset in a reduced-dimensional space. The clustering of points shows some separation but also a significant overlap, indicating that spam and ham messages share some common features.

Confusion Matrices for LDA and QDA

Confusion Matrix for LDA(Figure 2a)

- True Positives (TP) [spam correctly classified]: 39
- False Negatives (FN) [spam misclassified as ham]: 116
- False Positives (FP) [ham misclassified as spam]: 920
- True Negatives (TN) [ham correctly classified]: 31

Observations:

- The LDA model **misclassifies a large number of ham messages as spam** (FP = 920), indicating that it struggles with false positives.
- It also fails to correctly identify spam messages (low TP = 39), meaning spam detection performance is poor.

Confusion Matrix for QDA(Figure 2b)

- The confusion matrix is **identical** to the one from LDA:
  - 39 TP, 116 FN, 920 FP, 31 TN
- This means **QDA is performing the same as LDA** in this case.
- Since QDA allows for more flexible decision boundaries, its similarity in performance suggests that the dataset does not benefit from a more complex model.
- **Both LDA and QDA perform poorly for spam classification.**
  - They misclassify a significant number of ham messages as spam (FP = 920).
  - Their spam detection rate is low (only 39 correct predictions out of all spam messages).
- **LDA's linear boundary might be insufficient, but QDA does not improve performance.**
  - QDA allows for non-linear decision boundaries, but since the confusion matrices are identical, this suggests that the dataset's feature space does not support better separation.
- The classification task is challenging with LDA and QDA, and both models struggle to differentiate spam from ham.
- Given the high false positive rate, this model may not be practical for real-world spam detection without further improvements.

E. SMOTE

SMOTE [4] was applied with $k = 5$ nearest neighbors to generate synthetic samples (Listing 18). This method helps to balance the class distribution by creating synthetic examples rather than over-sampling with replacement.

```r
# Load necessary libraries
library(readr)       # For reading the CSV file
library(caret)       # For modeling and evaluation
library(smotefamily) # For SMOTE

# Load the data
data <- read_csv("/home/fenrir/T l chargements/sms+spam+collection/
    SMSSpamCollection.csv", col_names = c("Label", "Message"))

# Encode the labels as factors
data$Label <- as.factor(ifelse(data$Label == "spam", 1, 0))

# Split data into training and testing sets
set.seed(123)
trainIndex <- createDataPartition(data$Label, p = 0.7, list = FALSE)
trainData <- data[trainIndex,]
testData <- data[-trainIndex,]

# Apply SMOTE
smote_data <- SMOTE(X = trainData[, -ncol(trainData)], Y = trainData$
    Label, K = 5, perc.over = 100, perc.under = 100)

# Verify the results
str(smote_data)
```

Listing 18: Using SMOTE to generate synthetic samples

After applying SMOTE, the dataset characteristics are as follows:

- Total number of observations increased from 3,875 to 6,460.
- The number of synthetic observations added was 2,585.
- Original number of observations in the minority class (positive class): 517.
- Original number of observations in the majority class (negative class): 3,358.

This adjustment led to a more balanced dataset, which is crucial for training machine learning models that perform well on both classes. The application of SMOTE has effectively balanced the dataset, providing a solid foundation for subsequent predictive modeling. Future work will involve evaluating the performance of classification models trained on this balanced dataset and comparing results with those obtained from the original imbalanced dataset.

The addition of synthetic samples should theoretically improve model performance by providing a more balanced perspective on both classes. It is expected that this adjustment will lead to better sensitivity (true positive rate) and specificity (true negative rate) in predictive modeling.

F. Comparison of LDA and QDA between original and SMOTEd data

The LDA and QDA analyses was performed again of the SMOTEd data in a similar manner as for the original.

LDA analysis

Prior Probabilities of Groups

Prior probabilities of groups:

Class	Probability
0	0.9956
1	0.0040
2	0.0004

This indicates a significant class imbalance, where Class 0 dominates the dataset, comprising nearly 99.56% of the samples.

Group Means

The group means represent the average values of each feature for the different classes:

Example features:

Feature	Mean (LD1)	Mean (LD2)
$X.ll$	6.36	−6.29
$X.re$	−28.21	−4.94
$X.harri$	5.95	−5.60
$X.award$	3.08	0.45
$X.million$	−1.88	−2.19

These values suggest that certain features significantly influence the classification process.

Coefficients of Linear Discriminants

The following are the coefficients for the first two discriminants (LD1 and LD2):

Feature Importance in LD1 and LD2:

Feature	LD1	LD2
$X.ll$	6.36	−6.29
$X.re$	−28.21	−4.94
$X.harri$	5.95	−5.60
$X.award$	3.08	0.45
$X.million$	−1.88	−2.19

Features with large absolute values in LD1 play the most significant role in distinguishing between classes.

Proportion of Variance Captured

The proportion of variance explained by each linear discriminant is:

Discriminant	Proportion of Variance
$LD1$	94.91%
$LD2$	5.09%

LD1 accounts for the vast majority of variance, meaning it is the most important axis for class separation.

Key Takeaways

- **Severe class imbalance:** Class 0 heavily dominates, potentially biasing the model.
- **Collinearity warning:** Some features are highly correlated, which can distort LDA results.
- **LD1 dominates variance:** Most of the separation occurs along LD1.

- **Feature importance:** Words like *award*, *million*, and *msg* strongly influence class separation.

QDA analysis

The confusion matrix for the Quadratic Discriminant Analysis (QDA) model provides insights into how well the model classifies messages into their respective categories (e.g., spam or non-spam). If the diagonal values (true positives for each class) are high, the model is performing well. Conversely, high off-diagonal values indicate misclassifications, which suggest that QDA may struggle with distinguishing between certain classes.

Accuracy and Performance Metrics

The model's classification performance is evaluated based on several key metrics:

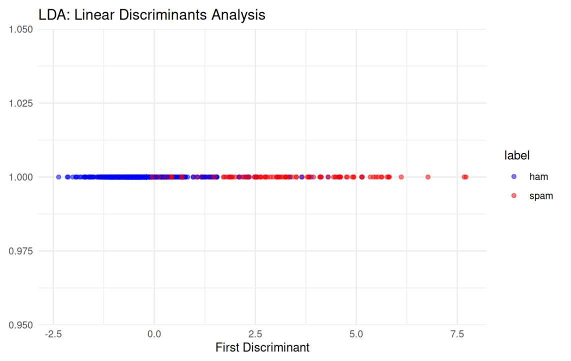
- **Overall Accuracy**: The proportion of correctly classified instances out of all instances.
- **Precision (Positive Predictive Value)**: Measures how many predicted spam messages are actually spam.
- **Recall (Sensitivity)**: Measures how many actual spam messages were correctly classified.

If QDA exhibits lower accuracy than LDA, it may be due to **overfitting**, as QDA estimates a separate covariance matrix for each class, which requires more data to be effective.

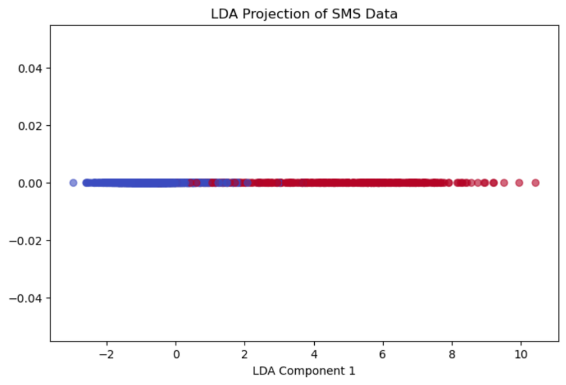
Class-Specific Performance

- If QDA struggles with minority classes (e.g., fewer spam messages), it may indicate class imbalance issues.
- False positive and false negative rates should be examined to determine whether QDA is more prone to misclassifying certain types of messages.

LDA analysis Comparison

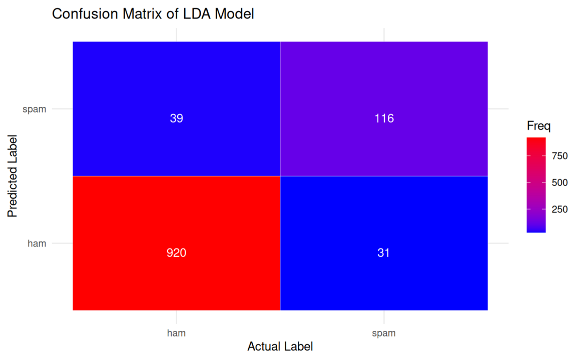


(a) Linear Discriminants Analysis on the original data

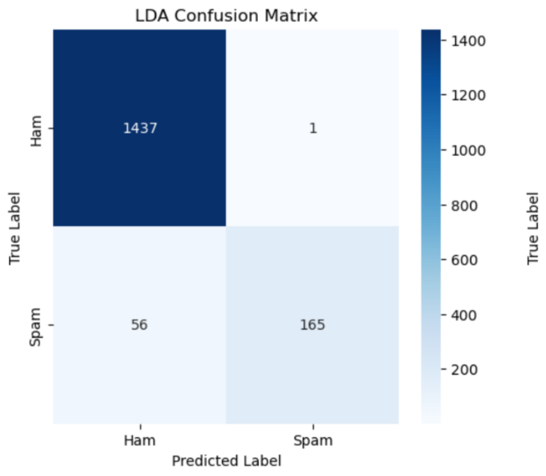


(b) Linear Discriminants Analysis on the SMOTEd data

Figure 3: Comparison of LDA between original and SMOTEd data



(a) LDA Confusion Matrix on the original data



(b) LDA Confusion Matrix on the SMOTEd data

Figure 4: Comparison of LDA Confusion Matrices between original and SMOTEd data

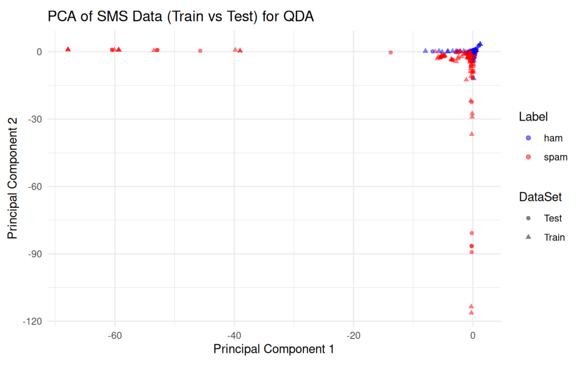
Feature	**LDA on Original Data**	**LDA on SMOTEed Data**
Spam Representation	Sparse	More balanced
Class Separability	Poor	Improved
Feature Space Spread	Minimal	More structured
Discriminant Axis Variance	Low	Higher

Table 1: Comparison of LDA performance on original vs. SMOTEed dataset.

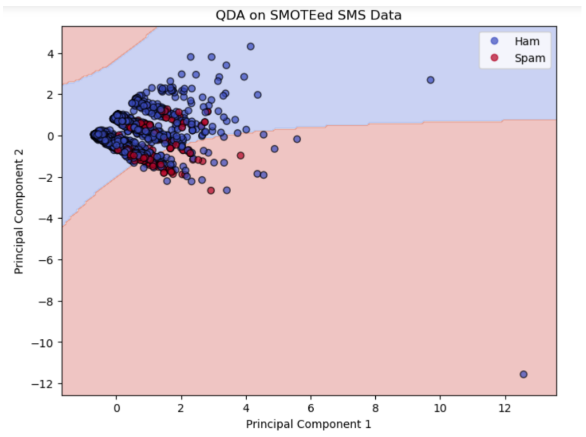
The effect of SMOTE on LDA projections is illustrated in Figure 3 . Without SMOTE, LDA fails to separate ham and spam messages effectively, as seen in the left plot (Figure 3a) where data points are clustered without significant variation. Spam messages are underrepresented, leading to a biased classification model. After applying SMOTE (Figure 3b), the spam and ham messages are better distributed along the discriminant axis, improving class separability. Table 1 summarizes the key differences, showing that SMOTE enhances the structure of the feature space and improves LDA's ability to distinguish between classes.

Figure 1 presents the confusion matrices for LDA before and after applying SMOTE. On the original dataset (Figure 1a), LDA exhibits a strong bias toward ham, leading to extremely poor spam detection. The model misclassifies a large number of spam emails as ham, resulting in low recall for spam detection. However, after applying SMOTE (Figure 1b), the classifier improves significantly, reducing misclassification errors and increasing its ability to detect spam emails correctly. Table 1 highlights these key improvements, showing how SMOTE enhances LDA's classification performance by addressing class imbalance.

QDA analysis Comparison

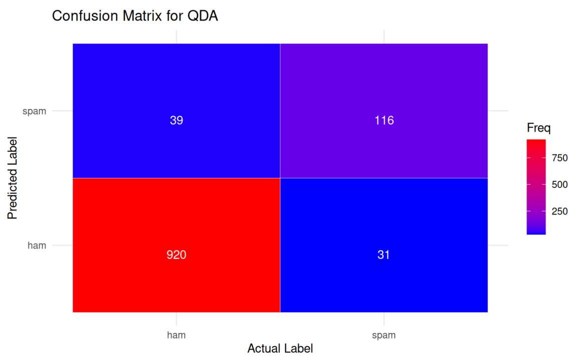


(a) Quadratic Discriminants Analysis on the original data

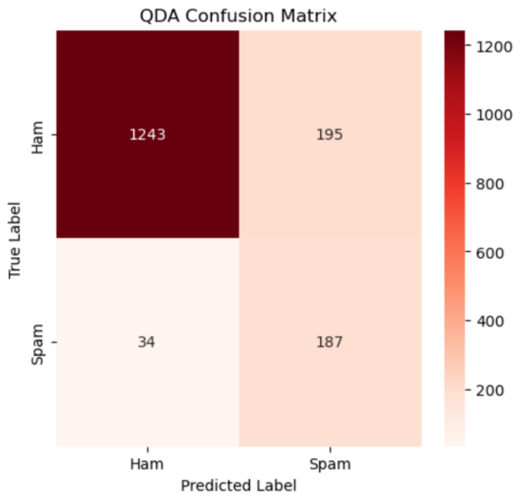


(b) Quadratic Analysis on the SMOTEd data

Figure 5: Comparison of QDA between original and SMOTEd data



(a) QDA Confusion Matrix on the original data



(b) QDA Confusion Matrix on the SMOTEd data

Figure 6: Comparison of QDA Confusion Matrices between original and SMOTEd data

In Figure 5a and Figure 5b) the performance of Quadratic Discriminant Analysis (QDA) is illustrated on different versions of the SMS dataset. The first graph (a) represents QDA applied to the original dataset before any resampling, while the second graph (b) represents QDA applied after balancing the dataset using the SMOTE technique.

In Figure 5a, the Principal Component Analysis (PCA) projection of the original dataset highlights a severe class imbalance. The spam messages (red) are significantly fewer compared to the ham messages (blue), indicating that the QDA model might struggle to effectively classify spam messages. Additionally, the clustering suggests that ham messages are tightly packed near the center, whereas spam messages are more dispersed, suggesting higher linguistic variability in spam messages. This imbalance may lead to overfitting, where the model becomes biased toward the majority class (ham) and underperforms in spam classification.

On the other hand, Figure 5b demonstrates the effect of SMOTE on improving QDA's decision boundary. The decision regions are now better defined, and the spam messages are more evenly distributed across the feature space. This suggests that QDA is now more capable of distinguishing spam messages. The reduction in overfitting is evident as spam messages occupy a larger feature space, allowing QDA to generalize better. However, some misclassifications still occur, indicating that further improvements, such as feature engineering or advanced classification models like Support Vector Machines (SVM) or deep learning, may be necessary.

The following table (Table 2) summarizes the differences between QDA applied to the original and SMOTEd datasets:

Feature	**QDA on Original Data**	**QDA on SMOTEed Data**
Spam vs. Ham Distribution	Highly imbalanced	More balanced
Decision Boundary	Poorly defined	More structured
Overfitting Risk	High (biased toward ham)	Lower (better spam detection)
Classification Performance	Likely poor on spam	Improved spam detection

Table 2: Comparison of QDA performance on original vs. SMOTEed dataset.

The confusion matrices in Figure 6 highlight the impact of SMOTE on QDA's classification performance. Without SMOTE, QDA exhibits a strong bias toward ham, misclassifying 920 spam messages as ham and correctly identifying only 39 spam emails. This leads to poor recall for spam detection. However, after applying SMOTE, QDA achieves a better balance, correctly classifying 187 spam emails while reducing spam misclassification to 34 cases. Although the number of ham emails misclassified as spam slightly increases, the overall spam detection performance significantly improves. As summarized in Table 2, SMOTE helps QDA generalize better by addressing class imbalance and enhancing spam detection accuracy.

G. Discussion and Conclusion

This study evaluated the performance of Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) for spam classification using the SMS Spam Collection dataset. The dataset was processed through TF-IDF transformation and Principal Component Analysis (PCA) to enhance feature representation. Additionally, SMOTE (Synthetic Minority Oversampling Technique) was applied to address class imbalance.

- LDA and QDA Performance: Both models struggled with spam classification, misclassifying a significant number of ham messages as spam. The confusion matrices for LDA and QDA were nearly identical, indicating that increasing model complexity (from linear to quadratic boundaries) did not significantly improve classification performance.

- Impact of PCA: The first few principal components (PC1–PC7) carried most of the discriminatory information, while later components contributed less. This suggests that dimensionality reduction was beneficial for improving efficiency but did not fully resolve classification challenges.

- Class Imbalance Effect: The dataset was highly imbalanced, with ham messages being far more frequent than spam messages. This skewness led to high false positive rates (ham misclassified as spam), which is problematic for real-world spam detection applications.

- SMOTE Effectiveness in LDA: SMOTE effectively reduces the class imbalance, enabling LDA to generalize better across both spam and ham messages. Spam detection performance improves significantly, with an increase in correctly classified spam emails and a decrease in misclassification errors. LDA becomes a more viable classifier for spam detection when class imbalance is addressed, demonstrating that data preprocessing techniques like SMOTE are crucial in improving model performance.

- SMOTE Effectiveness in QDA: The application of SMOTE significantly improves QDA's ability to classify spam messages by addressing class imbalance. Without SMOTE, QDA struggles to correctly classify spam due to the dominance of ham messages in the training set. Although SMOTE enhances performance, some misclassifications remain, suggesting that testing other models such as SVM, Random Forest, or Neural Networks could further improve classification accuracy.

Limitations

- Feature Representation: TF-IDF was used for text representation, but alternative approaches such as word embeddings (e.g., Word2Vec, BERT) could capture contextual meaning more effectively.

- Model Selection: LDA and QDA were evaluated, but other models such as logistic regression, support vector machines (SVMs), or deep learning architectures (LSTMs, transformers) could provide better spam detection performance.

- Handling Class Imbalance: While SMOTE helped balance the dataset, alternative techniques like cost-sensitive learning or ensemble methods could be explored to improve classification in imbalanced settings.

- Real-World Applicability: The high false positive rate in both LDA and QDA suggests that these models may not be suitable for production spam filters. Future work should focus on optimizing models that minimize both false positives and false negatives.

Therefore, Spam detection is a challenging classification task due to the evolving nature of spam messages and their overlap with legitimate messages. While LDA and QDA provide interpretable results, they are limited in their ability to distinguish spam effectively. Further exploration of advanced text processing techniques and modern machine learning models is necessary to develop a robust spam detection system.

References

[1] Jean Marc Freyermuth. *Do It Yourself 1 - Solution*. 2024-09-08. Homework document on class imbalance, LDA, QDA, and SMOTE in R.

[2] Ripley, B. D. *Modern Applied Statistics with S*. Springer, 2002.

[3] Almeida, T. A., Hidalgo, J. M. G.,& Yamakami, A. "SMS Spam Collection." `https://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`

[4] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. "SMOTE: Synthetic Minority Oversampling Technique." Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[5] Fisher, R. A. "The Use of Multiple Measurements in Taxonomic Problems." Annals of Eugenics, vol. 7, no. 2, pp. 179-188, 1936.

[6] Hastie, T., Tibshirani, R., & Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2009.

[7] Feinerer, I., Hornik, K., & Meyer, D. "Text Mining Infrastructure in R." Journal of Statistical Software, vol. 25, no. 5, pp. 1-54, 2008.

[8] Kuhn, M. "Building Predictive Models in R Using the caret Package." Journal of Statistical Software, vol. 28, no. 5, pp. 1-26, 2008.

[9] Salton, G., & Buckley, C. "Term-Weighting Approaches in Automatic Text Retrieval." Information Processing & Management, vol. 24, no. 5, pp. 513-523, 1988.

[10] Jolliffe, I. T. *Principal Component Analysis.* Springer, 2nd ed., 2002.

[11] Microsoft Corporation and Weston, S. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package.* R package version 1.0.17, 2022. `https://cran.r-project.org/package=doParallel`