Final Project

2025-06-04

```
train = read.table("spam-train.txt", sep = ",")
test = read.table("spam-test.txt", sep = ",")
train$V58 = as.factor(train$V58)
test$V58 = as.factor(test$V58)

train_errors = list()
test_errors = list()
```

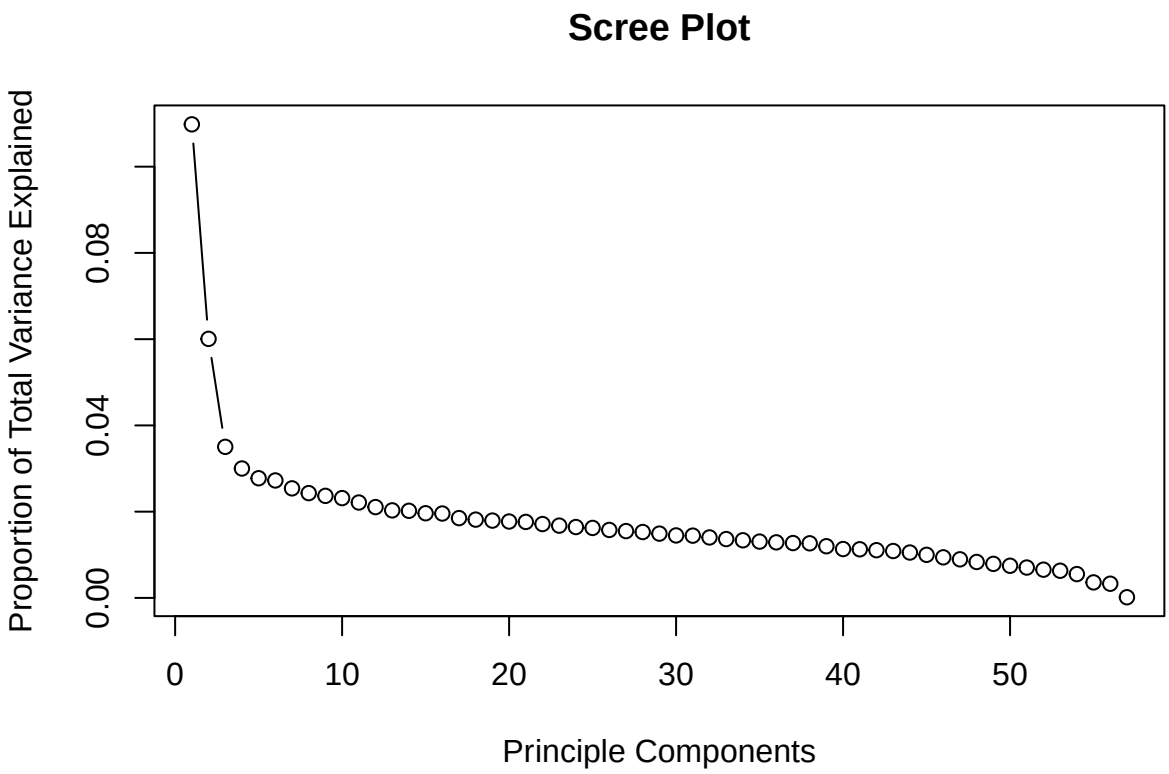
Q1

```
standardized_train = as.data.frame(scale(train[, 1:57]))
standardized_test = as.data.frame(scale(test[, 1:57]))
```

Q1A

```
pca_result = prcomp(standardized_train, center = FALSE, scale. = FALSE)
eigenvalues = pca_result$sdev^2
prop_var = eigenvalues / sum(eigenvalues)

plot(prop_var, type = "b", main = "Scree Plot",
     ylab = "Proportion of Total Variance Explained",
     xlab = "Principle Components")
```



Q1B Logistic Regression Model

```
log_reg = glm(train$V58 ~ ., data = as.data.frame(standardized_train), family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
p_vals = summary(log_reg)$coefficients[, 4]
significant_features = names(p_vals[p_vals <= 0.05 & names(p_vals) != "(Intercept)"])

train_probs = predict(log_reg, newdata = standardized_train, type = "response")
train_preds = ifelse(train_probs > 0.5, 1, 0)
train_error = mean(train_preds != train$V58)

test_probs = predict(log_reg, newdata = standardized_test,  type = "response")
test_preds = ifelse(test_probs > 0.5, 1, 0)
test_error = mean(test_preds != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error:", train_error, "\n", "Test Error:", test_error, "\n")
```

```
## Train Error: 0.07173133
##  Test Error: 0.07105606
```

```
print(significant_features)
```

```
##  [1] "V4"  "V5"  "V7"  "V8"  "V9"  "V11" "V16" "V17" "V19" "V20" "V21" "V23"
## [13] "V25" "V27" "V42" "V44" "V45" "V46" "V47" "V49" "V52" "V53" "V55" "V56"
## [25] "V57"
```

The logistic regression model seems to be quite accurate on predicting the standardized data, having little to no difference between the error in training predictions and test predictions.

Q1C LDA and QDA

```
#std lda
spam_std_lda = lda(train$V58 ~., data = standardized_train)

spam_std_predict_train = predict(spam_std_lda, standardized_train)
pred_val_std_lda_train = spam_std_predict_train$class
train_error = mean(pred_val_std_lda_train != train$V58)

spam_std_predict_test = predict(spam_std_lda, standardized_test)
pred_val_std_lda_test = spam_std_predict_test$class
test_error = mean(pred_val_std_lda_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("LDA Standardized Training Error :", train_error, "\n",
    "LDA Standardized Test Error:", test_error)
```

```
## LDA Standardized Training Error : 0.1017281
##  LDA Standardized Test Error: 0.1029987
```

```
#std qda
spam_std_qda = qda(train$V58 ~., data = standardized_train)

spam_std_predict_train_qda = predict(spam_std_qda, standardized_train)
pred_val_std_qda_train = spam_std_predict_train_qda$class
train_error = mean(pred_val_std_qda_train != train$V58)

spam_std_predict_test_qda = predict(spam_std_qda, standardized_test)
pred_val_std_qda_test = spam_std_predict_test_qda$class
test_error = mean(pred_val_std_qda_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("QDA Standardized Training Error:", train_error, "\n",
    "QDA Standardized Test Error:", test_error)
```

```
## QDA Standardized Training Error: 0.1786762
##  QDA Standardized Test Error: 0.1747066
```

Q1D Linear and NonLinear SVM

```
# standardized linear svm
svm_model = svm(train$V58 ~ ., data = standardized_train, kernel = "linear")

pred_train = predict(svm_model, newdata = standardized_train)
```

```r
train_error = mean(pred_train != train$V58)

pred_test = predict(svm_model, newdata = standardized_test)
test_error = mean(pred_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error", train_error, "\n", "Test Error:", test_error)
```

```
## Train Error 0.06488425
##  Test Error: 0.07170795
```

```r
# stanardized radial svm
radial_model = svm(train$V58 ~ ., data = standardized_train, kernel = "radial")

pred_train = predict(radial_model, newdata = standardized_train)
train_error = mean(pred_train != train$V58)

pred_test = predict(radial_model, newdata = standardized_test)
test_error = mean(pred_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error", train_error, "\n", "Test Error:", test_error)
```

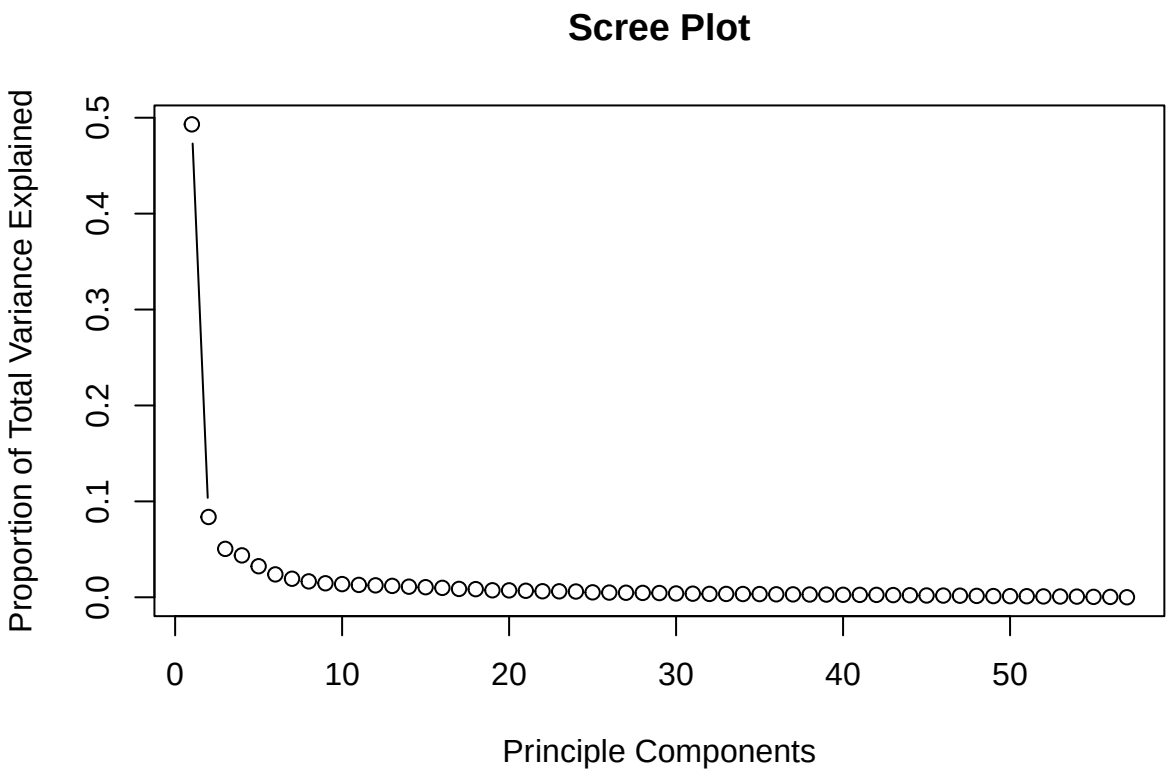
```
## Train Error 0.05151614
##  Test Error: 0.06453716
```

Q2

```r
log_transformed_train = as.data.frame(log1p(train[, 1:57]))
log_transformed_test = as.data.frame(log1p(test[, 1:57]))
```

```r
pca_result = prcomp(log_transformed_train, center = TRUE, scale. = FALSE)
eigenvalues = pca_result$sdev^2
prop_var = eigenvalues / sum(eigenvalues)

plot(prop_var, type = "b", main = "Scree Plot",
     ylab = "Proportion of Total Variance Explained",
     xlab = "Principle Components")
```



Q2A Logistic Regression Model

```
log_reg = glm(train$V58 ~ ., data = as.data.frame(log_transformed_train),
              family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
p_vals = summary(log_reg)$coefficients[, 4]
significant_features = names(p_vals[p_vals <= 0.05 & names(p_vals) != "(Intercept)"])

train_probs = predict(log_reg, newdata = log_transformed_train, type = "response")
train_preds = ifelse(train_probs > 0.5, 1, 0)
train_error = mean(train_preds != train$V58)

test_probs = predict(log_reg, newdata = log_transformed_test,  type = "response")
test_preds = ifelse(test_probs > 0.5, 1, 0)
test_error = mean(test_preds != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error:", train_error, "\n", "Test Error:", test_error, "\n")
```

```
## Train Error: 0.05771112
##  Test Error: 0.05671447
```

```
print(significant_features)
```

```
##  [1] "V5"  "V7"  "V8"  "V11" "V13" "V16" "V17" "V20" "V21" "V23" "V24" "V25"
## [13] "V27" "V28" "V33" "V35" "V37" "V42" "V43" "V45" "V46" "V49" "V52" "V53"
## [25] "V57"
```

Again, the logistic regression model seems to be quite accurate on predicting the log-transformed data, having little to no difference between the error in training predictions and test predictions, and this time having a lower error in both train and test predictions compared to the standardized data.

Q2B LDA and QDA

```
#log lda
spam_log_lda = lda(train$V58 ~., data = log_transformed_train)

spam_log_predict_train = predict(spam_log_lda, log_transformed_train)
pred_val_log_lda_train = spam_log_predict_train$class
train_error = mean(pred_val_log_lda_train != train$V58)

spam_log_predict_test = predict(spam_log_lda, log_transformed_test)
pred_val_log_lda_test = spam_log_predict_test$class
test_error = mean(pred_val_log_lda_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("LDA Log-transformed Training Error:", train_error, "\n",
    "LDA Log-transformed Test Error:", test_error)
```

```
## LDA Log-transformed Training Error: 0.06031953
##  LDA Log-transformed Test Error: 0.06518905
```

```
#log qda
spam_log_qda = qda(train$V58 ~., data = log_transformed_train)

spam_log_predict_train_qda = predict(spam_log_qda, log_transformed_train)
pred_val_log_qda_train = spam_log_predict_train_qda$class
train_error = mean(pred_val_log_qda_train != train$V58)

spam_log_predict_test_qda = predict(spam_log_qda, log_transformed_test)
pred_val_log_qda_test = spam_log_predict_test_qda$class
test_error = mean(pred_val_log_qda_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("QDA Log-transformed Training Error:", train_error, "\n",
    "QDA Log-transformed Test Error:", test_error)
```

```
## QDA Log-transformed Training Error: 0.1587871
##  QDA Log-transformed Test Error: 0.1571056
```

Q2C Linear and NonLinear SVM

```
# log-transformed linear svm
svm_model = svm(train$V58 ~ ., data = log_transformed_train, kernel = "linear")

pred_train = predict(svm_model, newdata = log_transformed_train)
train_error = mean(pred_train != train$V58)

pred_test = predict(svm_model, newdata = log_transformed_test)
test_error = mean(pred_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error", train_error, "\n", "Test Error:", test_error)
```

```
## Train Error 0.05412455
##  Test Error: 0.05149935
```

```
# log-transformed radial svm
radial_model = svm(train$V58 ~ ., data = log_transformed_train, kernel = "radial")

pred_train = predict(radial_model, newdata = log_transformed_train)
train_error = mean(pred_train != train$V58)

pred_test = predict(radial_model, newdata = log_transformed_test)
test_error = mean(pred_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error", train_error, "\n", "Test Error:", test_error)
```

```
## Train Error 0.03880013
##  Test Error: 0.04498044
```

Q3

```
discretized_train = as.data.frame((train[1:57] > 0) * 1)
discretized_test = as.data.frame((test[1:57] > 0) * 1)
```

Q3A Logistic Regression Model

```
log_reg = glm(train$V58 ~ ., data = as.data.frame(discretized_train),
              family = binomial)
p_vals = summary(log_reg)$coefficients[, 4]
significant_features = names(p_vals[p_vals <= 0.05 & names(p_vals) != "(Intercept)"])

train_probs = predict(log_reg, newdata = discretized_train, type = "response")
train_preds = ifelse(train_probs > 0.5, 1, 0)
train_error = mean(train_preds != train$V58)

test_probs = predict(log_reg, newdata = discretized_test,  type = "response")
```

```
test_preds = ifelse(test_probs > 0.5, 1, 0)
test_error = mean(test_preds != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error:", train_error, "\n", "Test Error:", test_error, "\n")
```

```
## Train Error: 0.05705902
##  Test Error: 0.08083442
```

```
print(significant_features)
```

```
##  [1] "V5"  "V7"  "V8"  "V10" "V11" "V13" "V14" "V15" "V16" "V17" "V18" "V20"
## [13] "V21" "V23" "V24" "V25" "V27" "V28" "V37" "V42" "V43" "V44" "V45" "V46"
## [25] "V48" "V52" "V53" "V54"
```

This time around, the logistic regression model seems to be quite accurate on predicting the discretized data. However, there seems to be a small difference in the training and test error which seems to lead to a small bit of overfitting in the model. This difference could be attributed to the discretization not being a complete representation of the data because of how it was transformed, with there only being 0's and 1's and no in-between.

Q3D Linear and NonLinear SVM

```
# discretized linear svm
svm_model = svm(train$V58 ~ ., data = discretized_train, kernel = "linear")
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'V55' and 'V56' and 'V57' constant. Cannot scale data.
```

```
pred_train = predict(svm_model, newdata = discretized_train)
train_error = mean(pred_train != train$V58)

pred_test = predict(svm_model, newdata = discretized_test)
test_error = mean(pred_test != test$V58)

train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error", train_error, "\n", "Test Error:", test_error)
```

```
## Train Error 0.06031953
##  Test Error: 0.07431551
```

```
# discretized radial svm
radial_model = svm(train$V58 ~ ., data = discretized_train, kernel = "radial")
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'V55' and 'V56' and 'V57' constant. Cannot scale data.
```

```
pred_train = predict(radial_model, newdata = discretized_train)
train_error = mean(pred_train != train$V58)

pred_test = predict(radial_model, newdata = discretized_test)
test_error = mean(pred_test != test$V58)
```

```r
train_errors = append(train_errors, train_error)
test_errors = append(test_errors, test_error)

cat("Train Error", train_error, "\n", "Test Error:", test_error)
```

```
## Train Error 0.06162374
##  Test Error: 0.0756193
```

Summary Table

	Train	Test
Standardized.Log.Reg	0.072	0.071
Standardized.LDA	0.102	0.103
Standardized.QDA	0.179	0.175
Standardized.Linear.SVM	0.065	0.072
Standardized.Radial.SVM	0.052	0.065
Log.Transformed.Log.Reg	0.058	0.057
Log.Transformed.LDA	0.060	0.065
Log.Transformed.QDA	0.159	0.157
Log.Transformed.Linear.SVM	0.054	0.051
Log.Transformed.Radial.SVM	0.039	0.045
Discretized.Log.Reg	0.057	0.081
Discretized.Linear.SVM	0.060	0.074
Discretized.Radial.SVM	0.062	0.076

Recommended Model

```r
#Bagging and Random Forest for std df
train$V58 = as.factor(train$V58)
test$V58 = as.factor(test$V58)

bag.spam = randomForest(train$V58 ~., data = standardized_train,
                        mtry= 57, importance = TRUE, ntree = 100)

yhat.bag = predict(bag.spam, newdata = standardized_test)
Error.bag = mean(yhat.bag != test$V58)

RF.spam = randomForest(train$V58 ~., data = standardized_train,
                       mtry = 8, importance = TRUE, ntree = 100)
yhat.RF = predict(RF.spam, newdata = standardized_test)
Error.RF = mean(yhat.RF != test$V58)

#Bagging and Random Forest for log df
bag.spam.log = randomForest(train$V58 ~., data = log_transformed_train,
                        mtry= 57, importance = TRUE, ntree = 100)

yhat.bag.log = predict(bag.spam.log, newdata = log_transformed_test)
Error.bag.log = mean(yhat.bag.log != test$V58)

RF.spam.log = randomForest(train$V58 ~., data = log_transformed_train,
                       mtry = 8, importance = TRUE, ntree = 100)
```

```r
yhat.RF.log = predict(RF.spam.log, newdata = log_transformed_test)
Error.RF.log = mean(yhat.RF.log != test$V58)

#bagging and randomforest for discretized df
bag.spam.disc = randomForest(train$V58 ~., data = discretized_train,
                        mtry= 57, importance = TRUE, ntree = 100)
yhat.bag.disc = predict(bag.spam.disc, newdata = discretized_test)
Error.bag.disc = mean(yhat.bag.disc != test$V58)
RF.spam.disc = randomForest(train$V58 ~., data = discretized_train,
                       mtry = 8, importance = TRUE, ntree = 100)
yhat.RF.disc = predict(RF.spam.disc, newdata = discretized_test)
Error.RF.disc = mean(yhat.RF.disc != test$V58)
```

Table 2: Classification Errors for Bagging/Random Forest

Data Type	Bagging Error	Random Forest Error
Standardized	0.0515	0.0352
Log-Transformed	0.0313	0.0241
Discretized	0.0469	0.0508

After testing Random Forest and Bagging on all three differently transformed datasets, we conclude that Random Forest applied to log-transformed data performed best in spam classification. The test error for log-transformed data is approximately 0.0241.