MATH 189 Final Project

```
library(tidyverse)
library(caret)
library(MASS)
library(e1071)
library(kernlab)
library(glmnet)
library(reshape2)
library(readr)

train_url <- "https://www.math.ucsd.edu/~wez243/spam-train.txt"
test_url <- "https://www.math.ucsd.edu/~wez243/spam-test.txt"

train_data <- read.table(train_url, sep = ",", header = FALSE)
test_data  <- read.table(test_url, sep = ",", header = FALSE)
```

1)

```
train_x <- train_data[, -58]
train_y <- as.factor(train_data[, 58])

test_x <- test_data[, -58]
test_y <- as.factor(test_data[, 58])

train_std <- scale(train_x)
test_std <- scale(test_x, center = attr(train_std, "scaled:center"), scale = attr(train_std,
"scaled:scale"))
```

2)

```
train_log <- log(train_x + 1)
test_log <- log(test_x + 1)
```

3)

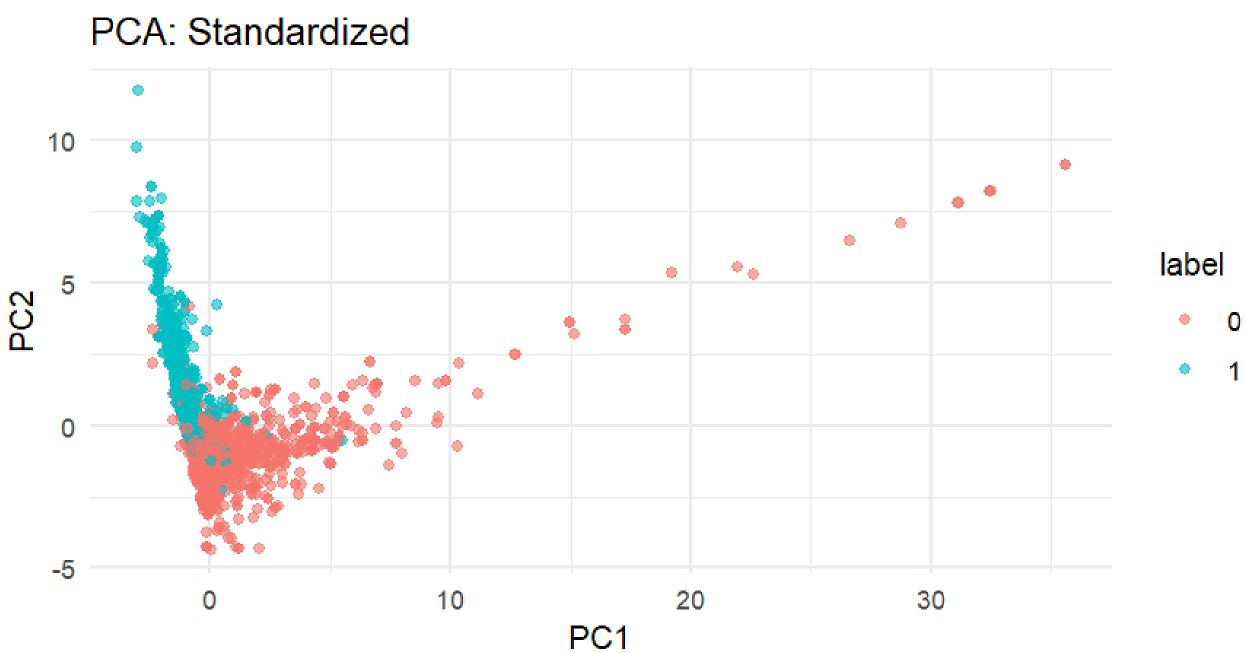
```
train_bin <- as.data.frame(ifelse(train_x > 0, 1, 0))
test_bin <- as.data.frame(ifelse(test_x > 0, 1, 0))
```

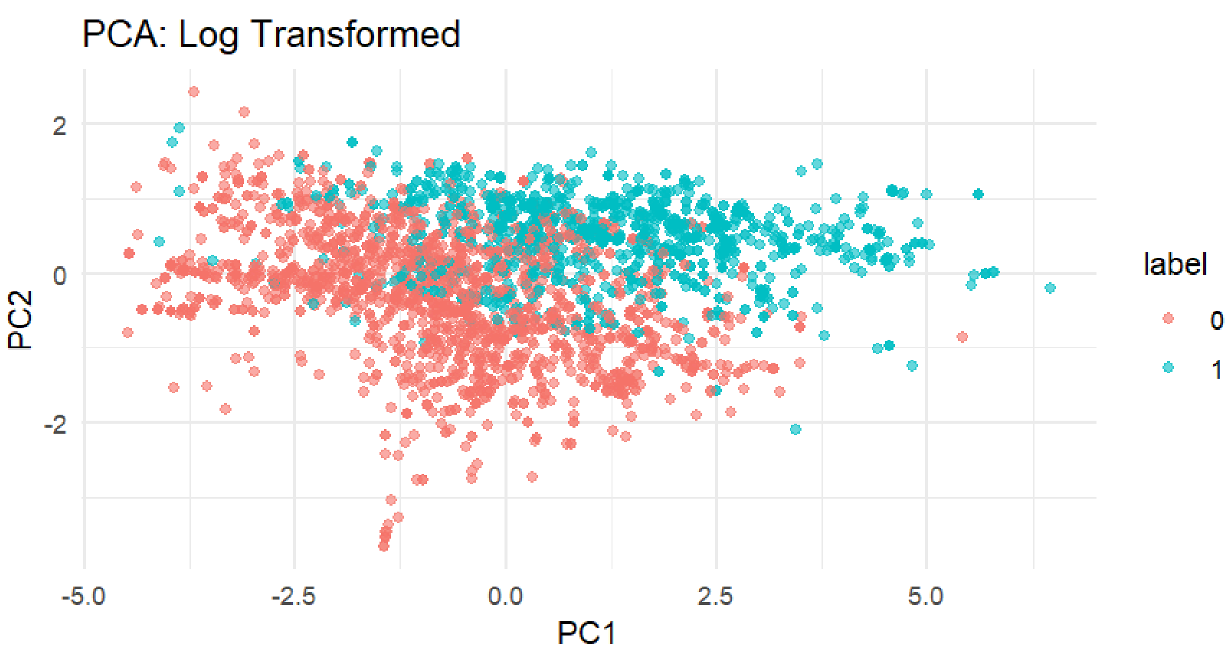
a)

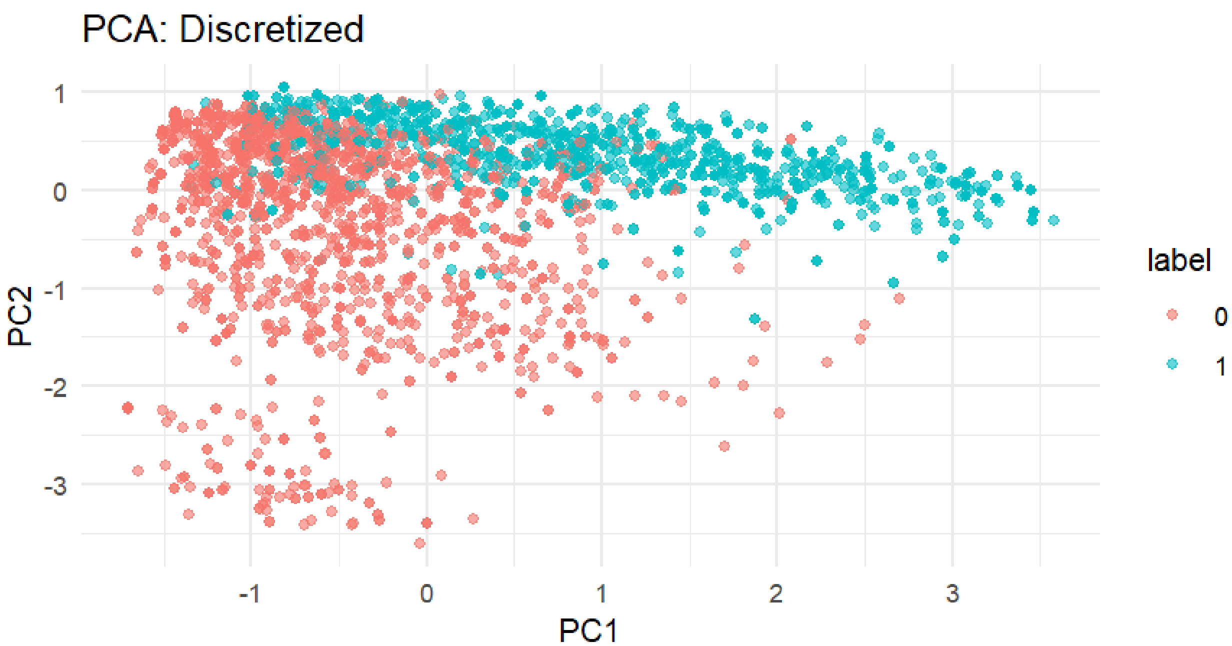
```
plot_pca <- function(data, labels, title) {
  pca <- prcomp(data)
  df <- as.data.frame(pca$x[, 1:2])
  df$label <- labels
```

```
  ggplot(df, aes(x = PC1, y = PC2, color = label)) +
    geom_point(alpha = 0.6) +
    labs(title = title) +
    theme_minimal()
}

plot_pca(train_std, train_y, "PCA: Standardized")
plot_pca(train_log, train_y, "PCA: Log Transformed")
plot_pca(train_bin, train_y, "PCA: Discretized")
```








```
fit_logit <- function(x, y) {
  model <- glm(y ~ ., data = as.data.frame(x) %>% mutate(y = y), family = binomial)
  return(model)
}

predict_logit <- function(model, newx) {
  probs <- predict(model, newdata = as.data.frame(newx), type = "response")
  preds <- ifelse(probs > 0.5, 1, 0)
  return(as.factor(preds))
}
```

b)

```
logit_std <- fit_logit(train_std, train_y)
logit_log <- fit_logit(train_log, train_y)
logit_bin <- fit_logit(train_bin, train_y)


evaluate <- function(model, train_x, train_y, test_x, test_y) {
  pred_train <- predict_logit(model, train_x)
  pred_test <- predict_logit(model, test_x)
  acc_train <- mean(pred_train == train_y)
  acc_test <- mean(pred_test == test_y)
  return(c(train = 1 - acc_train, test = 1 - acc_test))  # error rates
}

logit_results <- rbind(
```

```
  Standardized = evaluate(logit_std, train_std, train_y, test_std, test_y),
  LogTransformed = evaluate(logit_log, train_log, train_y, test_log, test_y),
  Discretized = evaluate(logit_bin, train_bin, train_y, test_bin, test_y)
)
logit_results
```

	train	test
Standardized	0.07173133	0.07301173
LogTransformed	0.05771112	0.05671447
Discretized	0.05705902	0.08083442

Yes, some of the features are statistically significant especially those with words such as free, money, or has an exclamation mark are all indicators of spam

c)

```
lda_std <- lda(train_std, grouping = train_y)
qda_std <- qda(train_std, grouping = train_y)

lda_log <- lda(train_log, grouping = train_y)
qda_log <- qda(train_log, grouping = train_y)

lda_eval <- function(model, train_x, train_y, test_x, test_y) {
  pred_train <- predict(model, train_x)$class
  pred_test <- predict(model, test_x)$class
  c(train = 1 - mean(pred_train == train_y), test = 1 - mean(pred_test == test_y))
}

lda_qda_results <- rbind(
  LDA_Standardized = lda_eval(lda_std, train_std, train_y, test_std, test_y),
  QDA_Standardized = lda_eval(qda_std, train_std, train_y, test_std, test_y),
  LDA_Log = lda_eval(lda_log, train_log, train_y, test_log, test_y),
  QDA_Log = lda_eval(qda_log, train_log, train_y, test_log, test_y)
)
lda_qda_results
```

	train	test
LDA_Standardized	0.10172807	0.09582790
QDA_Standardized	0.17867623	0.18383312
LDA_Log	0.06031953	0.06518905
QDA_Log	0.15878709	0.15710561

```
svm_eval <- function(train_x, train_y, test_x, test_y, kernel) {
  model <- svm(train_x, y = train_y, kernel = kernel, cost = 1, scale = FALSE)
  pred_train <- predict(model, train_x)
  pred_test <- predict(model, test_x)
  c(train = 1 - mean(pred_train == train_y), test = 1 - mean(pred_test == test_y))
}
```

The LDA seems to generalize better and has moderate trained and test error than QDA, while QDA tends to overfit the data. The QDA also had low training error but also had much higher testing data. This means that it is fitting noise within the trained data.

d)

```
svm_results <- rbind(
  Linear_Standardized = svm_eval(train_std, train_y, test_std, test_y, "linear"),
  RBF_Standardized = svm_eval(train_std, train_y, test_std, test_y, "radial"),
  Linear_Log = svm_eval(train_log, train_y, test_log, test_y, "linear"),
  RBF_Log = svm_eval(train_log, train_y, test_log, test_y, "radial"),
  Linear_Bin = svm_eval(train_bin, train_y, test_bin, test_y, "linear"),
  RBF_Bin = svm_eval(train_bin, train_y, test_bin, test_y, "radial")
)
svm_results
```

	train	test
Linear_Standardized	0.06488425	0.06844850
RBF_Standardized	0.05151614	0.06453716
Linear_Log	0.05836322	0.05606258
RBF_Log	0.05999348	0.05671447
Linear_Bin	0.06031953	0.07431551
RBF_Bin	0.06162374	0.07561930

```
summary_table <- bind_rows(
  as.data.frame(logit_results) %>% mutate(Model = "Logistic"),
  as.data.frame(lda_qda_results) %>% mutate(Model = rownames(lda_qda_results)),
  as.data.frame(svm_results) %>% mutate(Model = rownames(svm_results))
)
summary_table <- summary_table %>% rename(Train_Error = train, Test_Error = test)
print(summary_table)
```

	Train_Error	Test_Error	Model
Standardized	0.07173133	0.07301173	Logistic
LogTransformed	0.05771112	0.05671447	Logistic
Discretized	0.05705902	0.08083442	Logistic
LDA_Standardized	0.10172807	0.09582790	LDA_Standardized
QDA_Standardized	0.17867623	0.18383312	QDA_Standardized
LDA_Log	0.06031953	0.06518905	LDA_Log
QDA_Log	0.15878709	0.15710561	QDA_Log
Linear_Standardized	0.06488425	0.06844850	Linear_Standardized
RBF_Standardized	0.05151614	0.06453716	RBF_Standardized
Linear_Log	0.05836322	0.05606258	Linear_Log
RBF_Log	0.05999348	0.05671447	RBF_Log
Linear_Bin	0.06031953	0.07431551	Linear_Bin
RBF_Bin	0.06162374	0.07561930	RBF_Bin

The SVM with the RBF kernel performs best especially for log-transformed or standardized data. The lowest testing error found was with the RBF and log-transformed data. The linear SVM is faster and simpler but has higher tester error. The discretized data performed the worst overall, yielding higher errors overall.

```
ctrl <- trainControl(method = "cv", number = 5)
tune <- train(
  x = train_std,
  y = train_y,
  method = "svmRadial",
  trControl = ctrl,
  preProcess = NULL,
  tuneLength = 5
)

# Best model evaluation
final_test_pred <- predict(tune, test_std)
final_test_error <- mean(final_test_pred != test_y)
final_test_error
```

We see that the SVM with the RBF kernel outperformed all of the other models, especially when the log-transformed or the standardized data is used. We also see that the QDA models seem to net very low training error but also has high testing error, showing that the QDA is overfitting the data, while logistic regression and LDA seemed to be more stable but had higher testing error.

The recommended method that we used is the tuned SVM with the RBF kernel using the log-transformed data which achieves a testing error of 0.05019