CS 498 AML Homework 1 Report

Part 1 (all performed with 80% and 20% train test split, iteration of 10):

1.A accuracy (Naive Bayes, treat 0 as it is):

The average accuracy over 10 times: 0.7418301

2.B accuracy (Naive Bayes, treat 0 in certain columns as NA):

The average accuracy over 10 times: 0.7098039

3.D accuracy (SVM, treat 0 as it is):

The average accuracy over 10 times: 0.7300654

Part 1 Code:

1.A (1.B has similar code):

```
trainIndex <- createDataPartition(dataset$V9, p = 0.8, list = FALSE)
train <- dataset[ trainIndex,]
test <- dataset[-trainIndex,]
numTrain <- nrow(train)

train.label0 <- as.matrix(train[which(train$V9 == 0),])[, 1:num.features]
train.label1 <- as.matrix(train[which(train$V9 == 1),])[, 1:num.features]

# Getting priors
train.label0.prior <- nrow(train.label0) / numTrain
train.label1.prior <- nrow(train.label1) / numTrain

# Assuming all features across samples are represented as normal model
# Getting mean and standard deviation
train.label0.expect <- colMeans(train.label0)
train.label1.expect <- colMeans(train.label1)

train.label0.std <- colSds(train.label0)
train.label1.std <- colSds(train.label1)

# Creating new column in
test$predict <- NA

# Making prediction
for (idx in 1:nrow(test)){

  x <- as.numeric(test[idx, 1:num.features])
  pred.zero <-sum(dnorm(x, mean = train.label0.expect, sd = train.label0.std, log = TRUE)) + log(train.label0.prior)
  pred.one <- sum(dnorm(x, mean = train.label1.expect, sd = train.label1.std, log = TRUE)) + log(train.label1.prior)
  if(pred.one >= pred.zero){
    test[idx, ]$predict <- 1
  } else {
    test[idx, ]$predict <- 0
  }
}

acc_list[each] <- sum(test$V9 == test$predict) / nrow(test)
```

1.D (Train/test split is similar to 1.A and 1.B, omitted for the space):

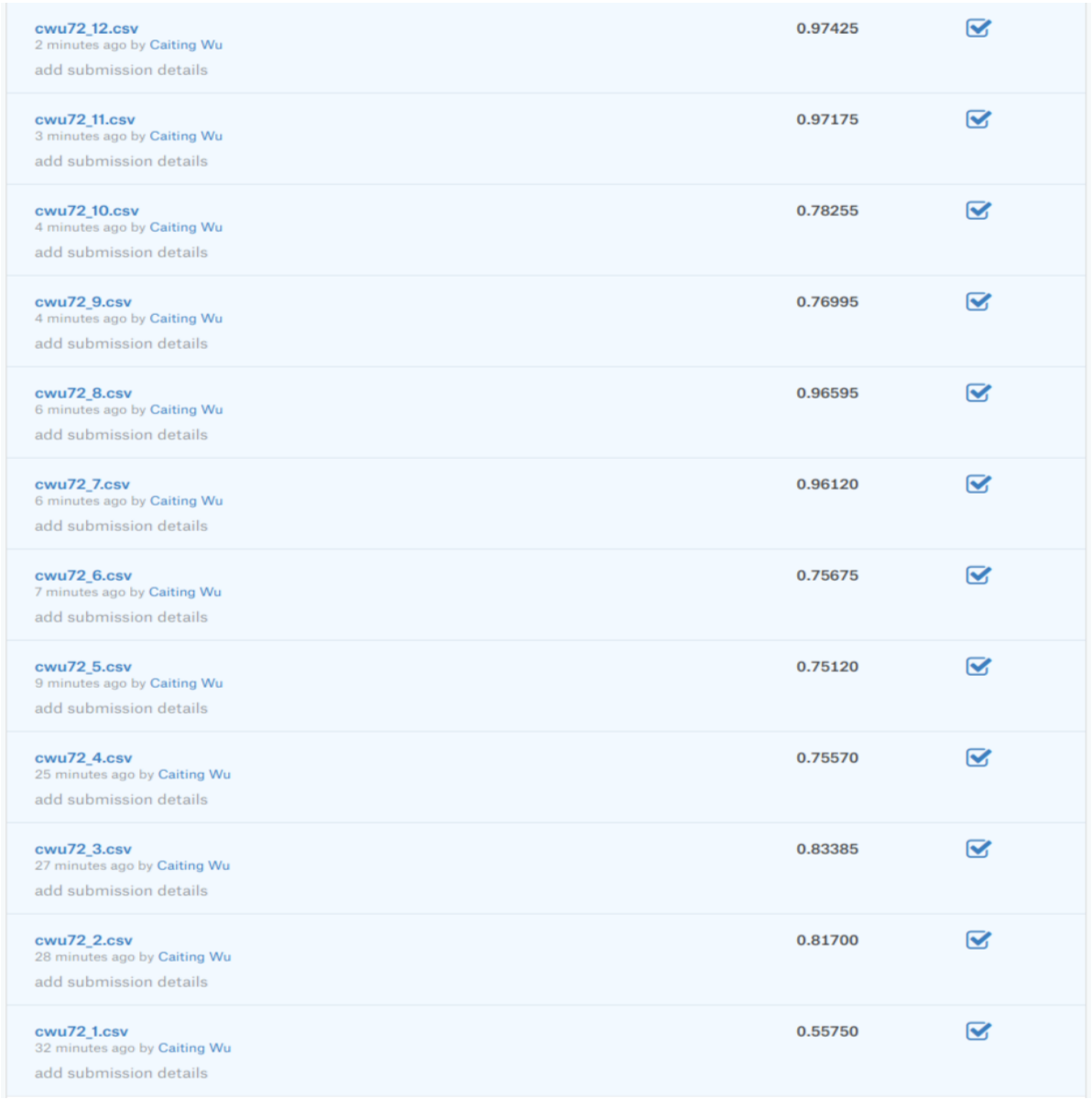
```
svm_model <- svmlight(train.feature, train.label, pathsvm = path)

prediction <- predict(svm_model, test.feature)

acc_list[each] <- sum(test.label == prediction$class) / num_test
```

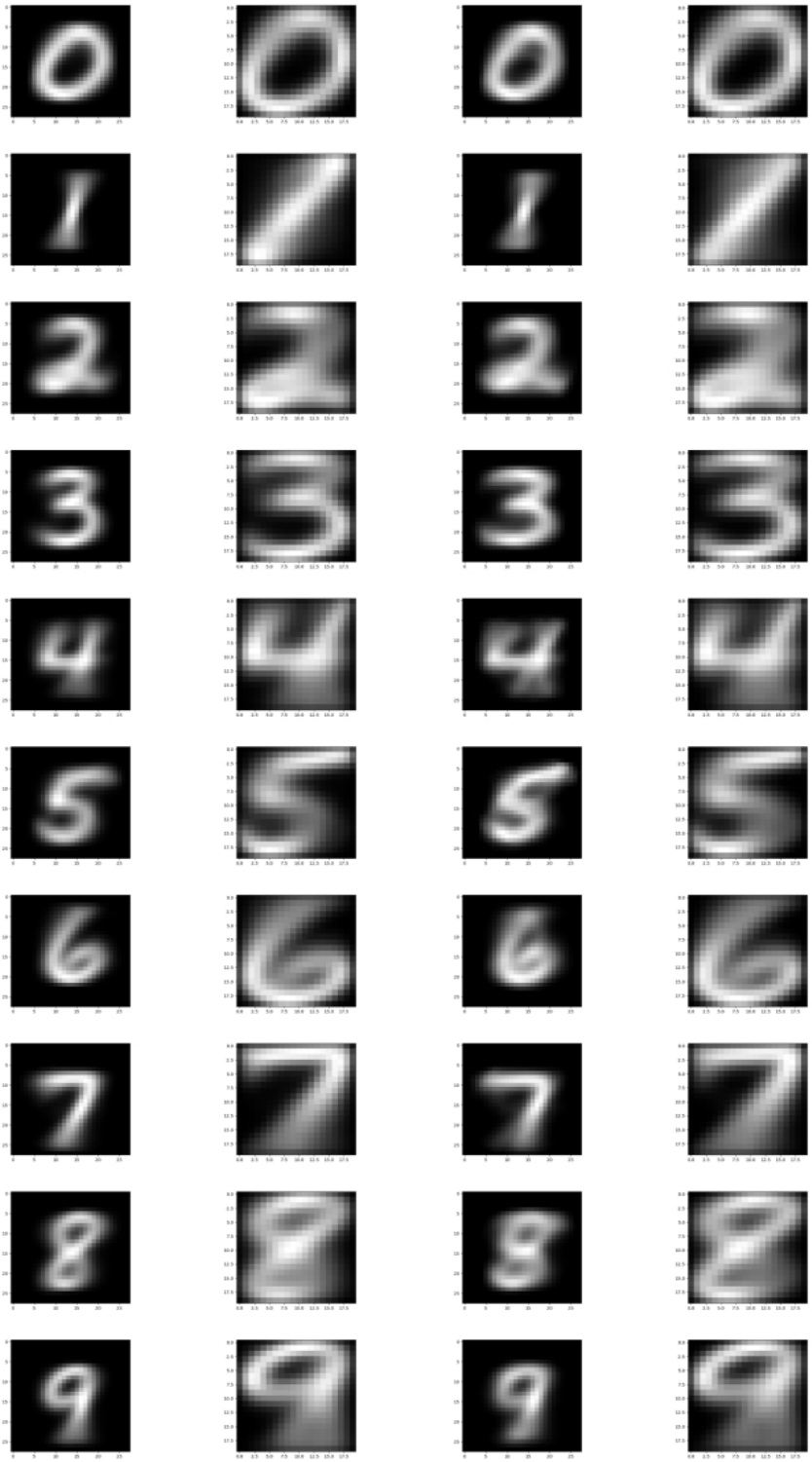
Part 2 (Table & Submission Screenshot):

#	Method	Testing Accuracy (%)
1	Gaussian + untouched	55.75
2	Gaussian + stretched	81.70
3	Bernoulli + untouched	83.385
4	Bernoulli + stretched	75.57
5	10 trees + 4 depth + untouched	75.12
6	10 trees + 4 depth + stretched	75.675
7	10 trees + 16 depth + untouched	96.12
8	10 trees + 16 depth + stretched	96.595
9	30 trees + 4 depth + untouched	76.995
10	30 trees + 4 depth + stretched	78.255
11	30 trees + 16 depth + untouched	97.175
12	30 trees + 16 depth + stretched	97.425



Part 2 (40 Pictures):

Gauss & untouch	Gauss + stretch	Bern + untouch	Bern + stretch



Part 2 (Codes):

```
# The script for CS 298 AML MP1 Part 2
import pandas as pd
import numpy as np
from skimage.transform import resize
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.ensemble import RandomForestClassifier
```

```
# Begin Naive Bayes Training
g_nb_1 = GaussianNB()
g_nb_2 = GaussianNB()
b_nb_1 = BernoulliNB()
b_nb_2 = BernoulliNB()

# For untouched
g_nb_1.fit(train_untouched, train_labels)
b_nb_1.fit(train_untouched, train_labels)
# For streched
g_nb_2.fit(train_scaled, train_labels)
b_nb_2.fit(train_scaled, train_labels)

# Validating
print('Gaussian + untouched validation acc:' , g_nb_1.score(val_untouched, val_labels))
print('Gaussian + stretched validation acc:' , g_nb_2.score(val_scaled, val_labels))
print('Bernoulli + untouched validation acc:' , b_nb_1.score(val_untouched, val_labels))
print('Bernoulli + stretched validation acc:' , b_nb_2.score(val_scaled, val_labels))
```

```
# Begin Random Forest Training
rfc_1 = RandomForestClassifier(n_estimators=10, max_depth=4)
rfc_2 = RandomForestClassifier(n_estimators=10, max_depth=4)
rfc_3 = RandomForestClassifier(n_estimators=10, max_depth=16)
rfc_4 = RandomForestClassifier(n_estimators=10, max_depth=16)
rfc_5 = RandomForestClassifier(n_estimators=30, max_depth=4)
rfc_6 = RandomForestClassifier(n_estimators=30, max_depth=4)
rfc_7 = RandomForestClassifier(n_estimators=30, max_depth=16)
rfc_8 = RandomForestClassifier(n_estimators=30, max_depth=16)

# Start Training
rfc_1.fit(total_untouched, total_labels)
rfc_2.fit(total_scaled, total_labels)
rfc_3.fit(total_untouched, total_labels)
rfc_4.fit(total_scaled, total_labels)
rfc_5.fit(total_untouched, total_labels)
rfc_6.fit(total_scaled, total_labels)
rfc_7.fit(total_untouched, total_labels)
rfc_8.fit(total_scaled, total_labels)

# Getting prediction and save it into .csv file with correct column label
# @NOTE: Same as before, still lazy
prediction = rfc_1.predict(test_untouched)
pred_df = pd.DataFrame(data={'Label': prediction})
pred_df.to_csv('cwu72_5.csv', index=True)
pred_df = pd.read_csv('cwu72_5.csv')
pred_df.columns = ['ImageId', 'Label']
pred_df.to_csv('cwu72_5.csv', index=False)
```