PartA

1.Processing the data: read csv and transfer to dataframe

2.Run the data for 10 times and shuffle the data each time so that we can have different training sets and testing sets each time.

3.Generate two matrix: yes matrix and no matrix, which will be used for future analysis of conditional probability for naïve bayes model.

4.Get the value of p(yes) and p(no)

5.Get the value of p(f1 | yes) …. P(yes) and p(f1 |no)….P(no)

6.See which value has larger value, then choose the one that has the larger value as our predicted result.

7.Test our data: traverse all the test data and get the average result, run 10 times and get the final overall accuracy.

The overall accuracy is about 0.74

PartB:

1.Processing the data: read csv and transfer to dataframe

2.Prepare our data: First, find all the zero value in column 3, 4, 6, 8
Second, transfer the zero value to NA (R representation)
Third, get the mean value of the column of other data
Fourth, replace the NA value data in the column with the mean value of other values in the current column.

3.Run the data for 10 times and shuffle the data each time so that we can have different training sets and testing sets each time.

4.Generate two matrix: yes matrix and no matrix, which will be used for future analysis of conditional probability for naïve bayes model.

5.Get the value of p(yes) and p(no)

6.Get the value of p(f1 | yes) …. P(yes) and p(f1 |no)….P(no)

7.See which value has larger value, then choose the one that has the larger value as our predicted result.

8.Test our data: traverse all the test data and get the average result, run 10 times and get the final overall accuracy.

The overall accuracy is about 0.81

PartD:

Predict the label using python sklearn.svm library

The overall accuracy is about 0.64

```r
############### Part A ###############


#run 10 times of split

#get the overall accuracy of running 10 times
average_accuracy <- 0

times <- 10
for (i in 1: times){

  #shuffle the data
  shuffled_data = data[sample(nrow(matrix_data)),]

  #get the size of data
  data_size <- nrow(data)

  #training size: 80% of data size
  training_size <- as.integer(data_size * 0.8)

  #number of yes and number of no
  num_of_yes = 0
  num_of_no = 0

  for(i in 1:training_size){
    if(shuffled_data[i, 9] == 1){
      num_of_yes <- num_of_yes + 1
    }
    else{
      num_of_no <- num_of_no + 1
    }
  }


  #generate the yes and no matricx
  yes_matrix <- c()
  no_matrix <- c()
```

```r
# p(yes) and p(no)
p_yes = num_of_yes / length(shuffled_data)
p_no = num_of_no / length(shuffled_data)


num_of_correct = 0


#####testing#####
for(i in training_size + 1: length((shuffled_data))){

  yes_score <- log(p_yes)
  no_score <- log(p_no)

  for(j in 1:8){
    yes_score <- yes_score + log(dnorm(as.matrix(shuffled_data[i, j]), mean = yes_mean_vec[j], sd = yes_sd_vec[j]))
    no_score <- no_score + log(dnorm(as.matrix(shuffled_data[i, j]), mean = no_mean_vec[j], sd = no_sd_vec[j]))
  }


  #estimated resulf from naive bayes
  estimated_result = 0
  if (yes_score > no_score){
    estimated_result = 1
  }
  else{
    estimated_result = 0
  }

  real_result = shuffled_data[i, 9]
  if(real_result == estimated_result){
    num_of_correct <- num_of_correct + 1
  }
}

accuracy = num_of_correct / length(shuffled_data)
print(accuracy)

average_accuracy <- average_accuracy + accuracy


overall_accuracy <- average_accuracy / 10
```

```r
############### Part B ###############

for(i in 1:nrow(matrix_data)){
  for(j in 1:8){
    if((j == 3 || j == 4 || j == 6 || j == 8) && matrix_data[i, j] == 0){
      matrix_data[i, j] <- NA
    }
  }
}



for(i in c(3, 4, 6, 8)){
  sum <- 0
  num <- 0
  for(j in 1:nrow(matrix_data)){
    if(is.na(matrix_data[j, i]) == FALSE){
      sum <- sum + matrix_data[j, i]
      num <- num + 1
    }
  }

  mean_val = sum / num
  print(mean_val)
  for(j in 1:nrow(matrix_data)){
    if(is.na(matrix_data[j, i]) == TRUE){
      matrix_data[j, i] <- mean_val
    }
  }
}


#run 10 times of split
```

```r
  #####testing#####
  for(i in training_size + 1: length((shuffled_data))){

    yes_score <- log(p_yes)
    no_score <- log(p_no)

    for(j in 1:8){
      yes_score <- yes_score + log(dnorm(as.matrix(shuffled_data[i, j]), mean = yes_mean_vec[j], sd = yes_sd_vec[j]))
      no_score <- no_score + log(dnorm(as.matrix(shuffled_data[i, j]), mean = no_mean_vec[j], sd = no_sd_vec[j]))
    }


    #estimated resulf from naive bayes
    estimated_result = 0
    if (yes_score > no_score){
      estimated_result = 1
    }
    else{
      estimated_result = 0
    }


    real_result = shuffled_data[i, 9]
    if(real_result == estimated_result){
      num_of_correct <- num_of_correct + 1
    }
  }

  accuracy = num_of_correct / length(shuffled_data)
  print(accuracy)


  average_accuracy <- average_accuracy + accuracy

}

overall_accuracy <- average_accuracy / 10
```

```
with open('pima-indians-diabetes.csv', 'r') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    size = 614

    count = 0

    for row in plots:
        if count < size:
            data.append(row[0:7])
            target.append(row[8])
        else:
            data2.append(row[0:7])
            target2.append(row[8])
        count += 1

model = SVC()
model.fit(data, target)
result = model.score(data2, target2)
print(result)#0.64
```

chongye2_12.csv
a few seconds ago by Chongye Wang
add submission details
0.82580

chongye2_11.csv
2 minutes ago by Chongye Wang
add submission details
0.97010

chongye2_10.csv
5 minutes ago by Chongye Wang
add submission details
0.61480

chongye2_9.csv
7 minutes ago by Chongye Wang
add submission details
0.78860

chongye2_8.csv
9 minutes ago by Chongye Wang
add submission details
0.75965

chongye2_7.csv
11 minutes ago by Chongye Wang
add submission details
0.96060

chongye2_6.csv — 0.54640
15 minutes ago by Chongye Wang
add submission details

chongye2_5.csv — 0.76820
17 minutes ago by Chongye Wang
add submission details

chongye2_4.csv — 0.67750
19 minutes ago by Chongye Wang
add submission details

chongye2_3.csv — 0.82740
22 minutes ago by Chongye Wang
add submission details

chongye2_2.csv — 0.66135
24 minutes ago by Chongye Wang
add submission details

chongye2_1.csv — 0.74010
27 minutes ago by Chongye Wang
add submission details

```
def get_middle(matrix):
    middle = []
    for i in range(4, 24):
        for j in range(4, 24):
            index = i * 28 + j
            middle.append(matrix[index])

    return np.array(middle).astype(np.float)


def stretch(matrix):
    reshaped_matrix = matrix.reshape(20, 20)
    left = 20
    right = -1
    up = 20
    down = -1
    for i in range(20):
        for j in range(20):
            if reshaped_matrix[i][j] != 0:
                if i < up: up = i
                if i > down: down = i
                if j < left: left = j
                if j > right: right = j

    height = down - up + 1
    width = right - left + 1
    stretched_matrix = reshaped_matrix[left:right + 1, up:down + 1]
    new_stretched = imresize(stretched_matrix, (20, 20))
    new_stretched = new_stretched.reshape(1, 400)[0]
```

```
#read test data
with open('val.csv', 'r') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    count = 0
    for row in data:
        if count == 0:
            count += 1
            continue
        curr = np.array(row[1:]).astype(np.float)
        test_data.append(curr)
        test_label.append(row[0])



train_data = np.array(train_data).astype(np.float)
train_label = np.array(train_label).astype(np.float)
test_data = np.array(test_data).astype(np.float)
test_label = np.array(test_label).astype(np.float)

middle_train_data = []
for i in range(len(train_data)):
    middle_train_data.append(get_middle(train_data[i]))
middle_train_data = np.array(middle_train_data).astype(np.float)

middle_test_data = []
for i in range(len(test_data)):
    middle_test_data.append(get_middle(test_data[i]))
middle_test_data = np.array(middle_test_data).astype(np.float)


stretched_matrix = []
for i in range(len(middle_train_data)):
    line = middle_train_data[i]
    stretched_matrix.append(stretch(line))
stretched_matrix = np.array(stretched_matrix).astype(np.float)
```

```python
#original v gaussian distribution
clf = GaussianNB()
clf.fit(middle_train_data, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('Original v Gaussian : ' + str(accuracy)) #0.7355


#original v bernoulli
clf1 = BernoulliNB()
clf1.fit(middle_train_data, train_label)
accuracy = clf1.score(middle_test_data, test_label)
print('Original v Bernoulli : ' + str(accuracy)) #0.8215



#stretched v gaussian dsitribution
clf2 = GaussianNB()
clf2.fit(stretched_matrix, train_label)
accuracy = clf2.score(middle_test_data, test_label)
print('Stretched v Gaussian : ' + str(accuracy)) #0.6505


#stretched v bernoulli
clf3 = BernoulliNB()
clf3.fit(stretched_matrix, train_label)
accuracy = clf3.score(middle_test_data, test_label)
print('Stretched v Bernoulli : ' + str(accuracy))#0.6795
```

```python
#stretched matrix
stretched_matrix = []
for i in range(len(middle_train_data)):
    line = middle_train_data[i]
    stretched_matrix.append(stretch(line))
stretched_matrix = np.array(stretched_matrix).astype(np.float)



##############################################################

#random forest
clf = RandomForestClassifier(n_estimators = 10, max_depth = 4)
clf.fit(middle_train_data, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('10 v 4 v original : ' + str(accuracy))  #0.7715

clf = RandomForestClassifier(n_estimators = 10, max_depth = 4)
clf.fit(stretched_matrix, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('10 v 4 v stretched : ' + str(accuracy))  #0.573



clf = RandomForestClassifier(n_estimators = 10, max_depth = 16)
clf.fit(middle_train_data, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('10 v 16 v original : ' + str(accuracy)) #0.968

clf = RandomForestClassifier(n_estimators = 10, max_depth = 16)
clf.fit(stretched_matrix, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('10 v 16 v stretched : ' + str(accuracy)) #0.7675
```

```python
clf = RandomForestClassifier(n_estimators = 10, max_depth = 16)
clf.fit(middle_train_data, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('10 v 16 v original : ' + str(accuracy)) #0.968

clf = RandomForestClassifier(n_estimators = 10, max_depth = 16)
clf.fit(stretched_matrix, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('10 v 16 v stretched : ' + str(accuracy)) #0.7675



clf = RandomForestClassifier(n_estimators = 30, max_depth = 4)
clf.fit(middle_train_data, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('30 v 4 v original : ' + str(accuracy)) #0.791

clf = RandomForestClassifier(n_estimators = 30, max_depth = 4)
clf.fit(stretched_matrix, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('30 v 4 v stretched : ' + str(accuracy)) #0.6165



clf = RandomForestClassifier(n_estimators = 30, max_depth = 16)
clf.fit(middle_train_data, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('30 v 16 v original : ' + str(accuracy)) #0.977

clf = RandomForestClassifier(n_estimators = 30, max_depth = 16)
clf.fit(stretched_matrix, train_label)
accuracy = clf.score(middle_test_data, test_label)
print('30 v 16 v stretched : ' + str(accuracy)) #0.8295
```