```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
```