ISYE 6501 HW2

January 2021

1 Question 3.1 KNN

1.1 Question 3.1a KNN K-fold CV

First I applied K-fold cross validation for KNN algorithm with 10 folds and k values from 0 to 20. The accuracy for different k values are shown below:

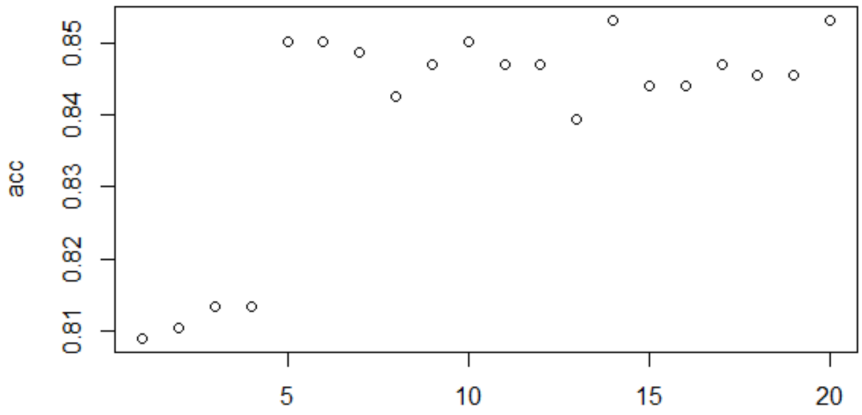


Table 1:

$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 8$	$k = 9$	$k = 10$
0.809	0.810	0.813	0.813	0.850	0.850	0.849	0.843	0.847	0.850
$k = 11$	$k = 12$	$k = 13$	$k = 14$	$k = 15$	$k = 16$	$k = 17$	$k = 18$	$k = 19$	$k = 20$
0.847	0.847	0.839	0.853	0.844	0.844	0.847	0.846	0.846	0.853

We could tell that the highest accuracy is when k=20 and k=14 with accuracy value of 0.853. But we can also tell that the accuracy for k=5 to k=20 are all within a small range [0.839, 0.853], which I wouldn't conclude that they are statistically significant different from each other. If I have to pick one model, I will pick k=20 or k=14 since they gave the highest accuracy.

1.2 Question 3.1b KNN K-fold CV

I'm using the createDataPartition function twice to split the data into 3 sets, training set with 70% of the data, validation set and testing set each contain 15% of the data. The accuracy values on validation set are close, many k values give the same accuracy.

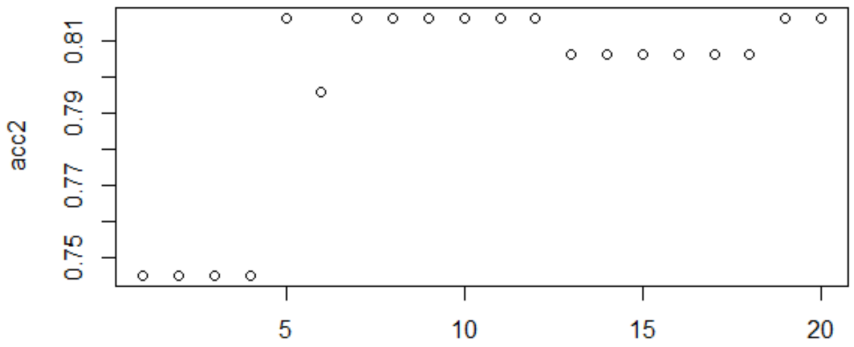


Table 2: Accuracy for K values – Testing data

$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
0.745	0.745	0.745	0.745	0.816	0.796	0.816	0.816	0.816	0.816
$k=11$	$k=12$	$k=13$	$k=14$	$k=15$	$k=16$	$k=17$	$k=18$	$k=19$	$k=20$
0.816	0.816	0.806	0.806	0.806	0.806	0.806	0.806	0.816	0.816

Then I fit the models into testing set, the plot of accuracy clearly shows that the model performs differently on the testing data from the validation data. k=6 performs the best on testing set with an accuracy of 0.888.

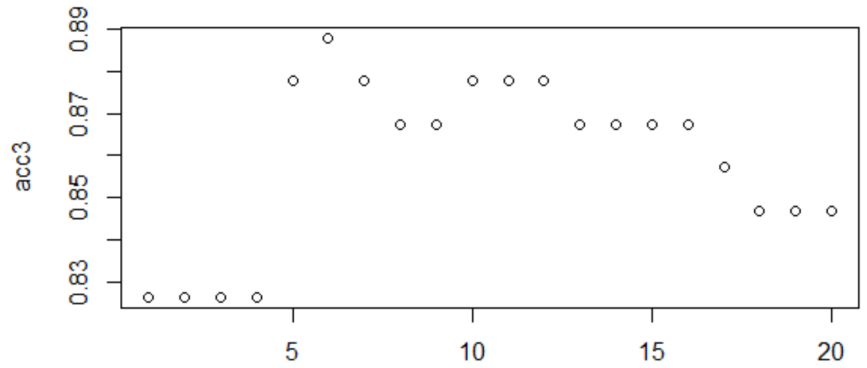


Table 3: Accuracy for K values – Testing data

$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
0.827	0.827	0.827	0.827	0.878	0.888	0.878	0.867	0.867	0.878
$k=11$	$k=12$	$k=13$	$k=14$	$k=15$	$k=16$	$k=17$	$k=18$	$k=19$	$k=20$
0.878	0.878	0.867	0.867	0.867	0.867	0.857	0.847	0.847	0.847

1.3 My R code is:

```
library(kknn)
library(caret)
set.seed(5)
data <- read.table("data 3.1/credit_card_data.txt",
                   stringsAsFactors = FALSE, header = FALSE)

# CV knn
check_accuracy = function(X){
  predicted <- rep(0,(nrow(data)))
  for (i in 1:nrow(data)){
    fit = cv.kknn(V11~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10, data, kcv = 10,
                  k=X, distance = 2, kernel = "optimal",ykernel = NULL, scale = TRUE)
    fit_df <- as.data.frame(fit)
    predicted[i] <- as.integer(fit_df[i,2]+0.5)
  }
  print(predicted)
  accuracy = sum(predicted == data[,11]) / nrow(data)
  return(accuracy)
}
acc <- rep(0,20)
for (X in 1:20){
  acc[X] = check_accuracy(X)
}
plot(acc)
print(acc)

# split data
datasplit <- createDataPartition(data[,11], 1, p=0.7, list=FALSE)
training_dataset  <- data[datasplit, ]
testing_validation <- data[-datasplit, ]
testing_validation_split <- createDataPartition(testing_validation[,11],
                                                1, p=0.5, list=FALSE)
validation_dataset <- testing_validation[testing_validation_split, ]
testing_dataset <- testing_validation[-testing_validation_split,]

# fit knn in training - validation set
check_accuracy2 = function(y){
  predicted2 <- rep(0,(nrow(validation_dataset)))
  knn_model = kknn(V11~., training_dataset, validation_dataset,
                       k=y, scale=TRUE)
  predicted2 = round(fitted(knn_model))
  print(predicted2)
  accuracy2 = sum(predicted2 == validation_dataset[,11]) / nrow(validation_dataset)
  return(accuracy2)
}
acc2 <- rep(0,20)
for (y in 1:20){
  acc2[y] = check_accuracy2(y)
}
plot(acc2)
print(acc2)
```

```
# fit knn in training - testing set
check_accuracy3 = function(z){
  predicted3 <- rep(0,(nrow(testing_dataset)))
  knn_model3 = kknn(V11~., training_dataset, testing_dataset,
                    k=z, scale=TRUE)
  predicted3 = round(fitted(knn_model3))
  print(predicted3)
  accuracy3 = sum(predicted3 == testing_dataset[,11]) / nrow(testing_dataset)
  return(accuracy3)
}
acc3 <- rep(0,20)
for (z in 1:20){
  acc3[z] = check_accuracy3(z)
}
plot(acc3)
print(acc3)
```

2 Question 4.1

An example would be a chain restaurant delivery method. Say there are few restaurants of the same chain in a metro area, we need to decide the area of responsibility for delivery of each restaurant. The key predictor would be route time, and route distance may also be considered.

3 Question 4.2 Kmeans Clustering

I have plotted several graphs to review the nature of the iris data. It seems that Petal Length and Petal Width are strongly positively correlated, which make sense, most larger Petals are both longer and wider than smaller Petals. But Sepal, on the other hand, do not show strong correlation for its length and width.

The relationship between Petal and Sepal is interesting. Looking at Petal Length in particular, When Petal Length is greater than 2, it is positively correlated with Sepal Width and Length. However, when Petal Length is less than 2, there is no evidence of correlation between Petal Length and Sepal. Similarly for Petal Width, when Petal Width is greater than 1 there is a clearly positive correlation, but when it is less than 1, there is no evidence of correlation between Petal Width and Sepal. What triggered these gaps?

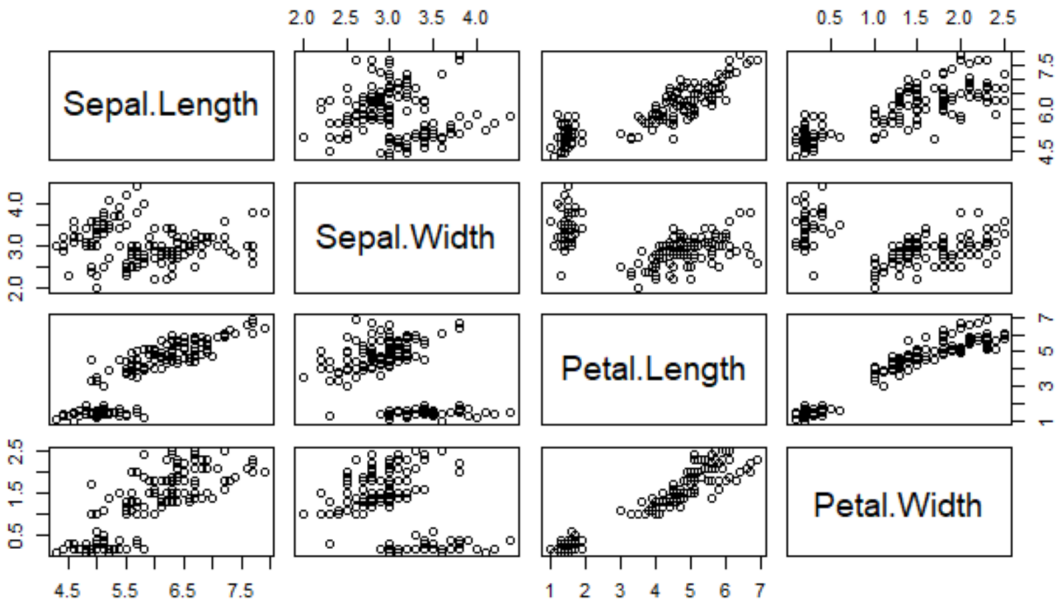


Figure 1: Relationship between each predictors

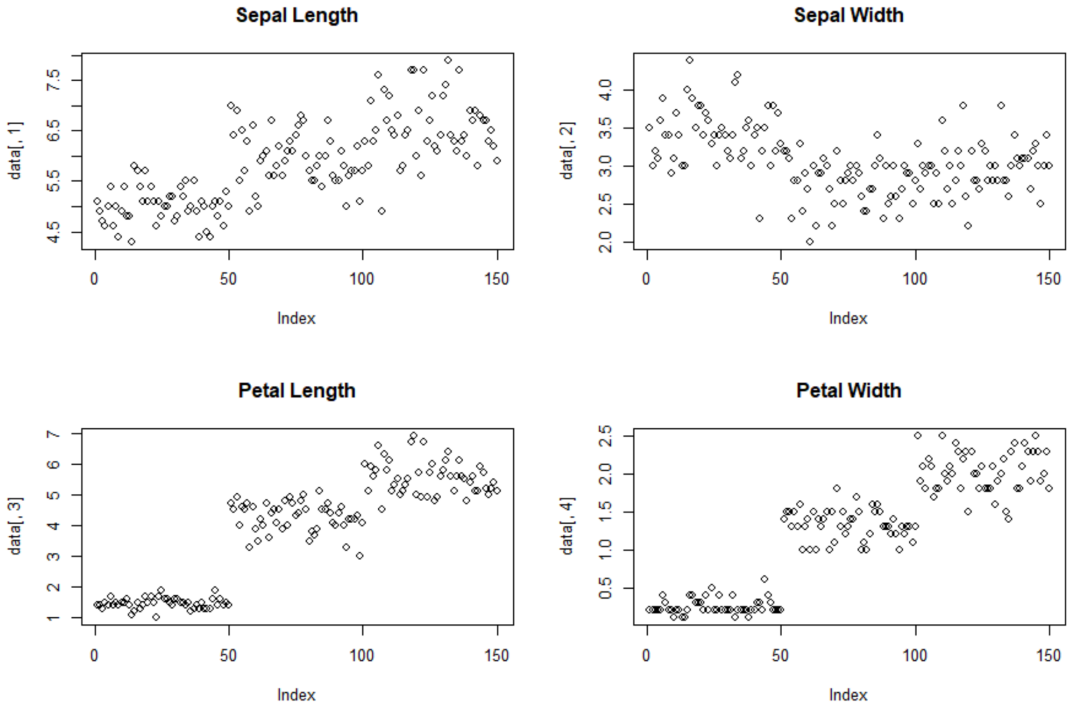
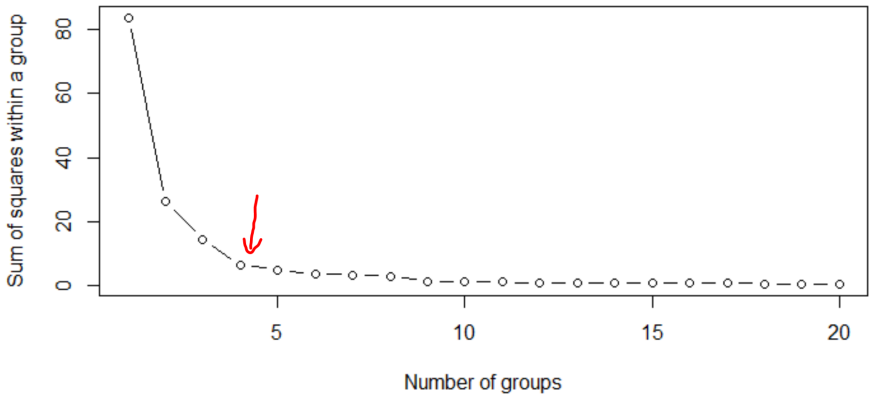


Figure 2: Review each predictor

By reviewing scatter plots of each predictor on it's own (Figure 2), I find that the Petal length and Petal Width dots formed 3 clusters in their corresponding plots. This may suggest that these 2 predictors have more impact on the model. And the data on the smaller end (both Petal Length and Width) shows a "flat" distribution, which means that for about 50 data points, their petal length/width are about the same (2 predictors are not necessarily for the same data point).

3.1 Question 4.2 Model

First, I have split the data into 2 sets, training set which contains 80% of the data, and testing set which contains the rest. Then, I run the Kmean Clustering algorithm on all predictors for k=1 to 20 using the training set. And the "elbow diagram" suggests that k=4 is the best fit.

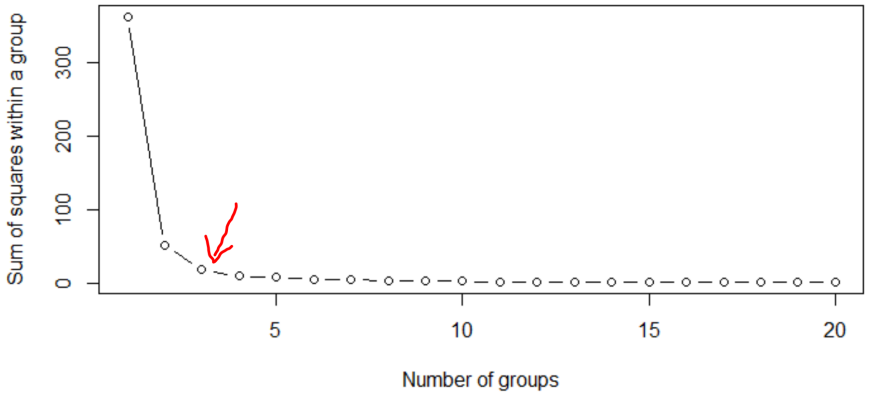


So I fit the model to training set with k=4 the compactness of this model is 92.3%.

When apply this model to the testing data, the compactness of this model is 92.6%.

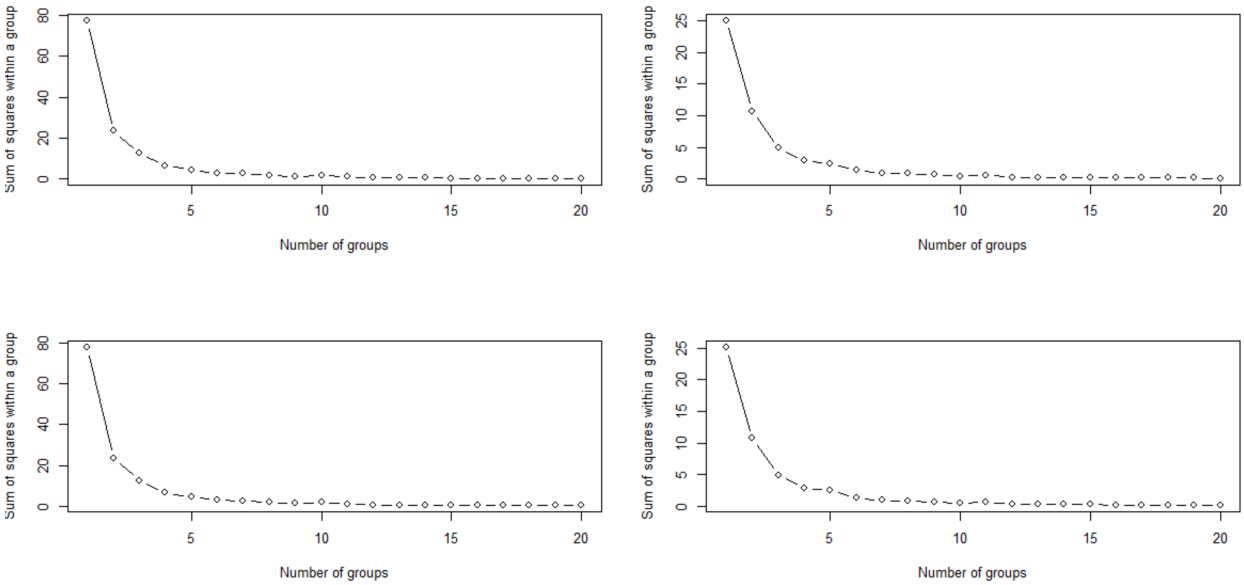
```
K-means clustering with 4 clusters of sizes 40, 11, 36, 33

Cluster means:
      [,1]
1 5.732500
2 7.436364
3 4.866667
4 6.506061

Clustering vector:
  [1] 3 3 3 3 3 3 3 1 3 3 3 1 1 1 3 1 3 1 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 3
 [40] 3 2 4 4 1 4 1 3 4 3 1 1 1 4 1 1 4 1 1 1 4 1 4 4 4 4 1 1 1 1 4 1 1 1 1 3 1 1
 [79] 4 1 1 2 4 2 3 2 4 2 4 4 4 1 1 4 4 2 1 4 1 2 4 4 1 4 2 2 2 4 4 2 4 1 4 4 1 4 4
[118] 4 4 1

Within cluster sum of squares by cluster:
[1] 1.7677500 0.8854545 2.2400000 1.5787879
 (between_SS / total_SS =  92.3 %)
```

Remember that the Petal predictors seems distributed in clusters. So I modify my model to only include the 2 Petal predictors. And the elbow diagram shows that k=3 is the turning point. And when fit the model to training set, compactness of this model is 94.8%. Fit to testing set gives a compactness of 95%.



Then I did a serious of analysis for picking different predictors and draw elbow diagram for each combination. The elbow diagrams for them are either not so smooth, or has a large turning point.

- Sepal Length, Sepal Width, Petal Length
- Sepal Width, Petal Length, Petal Width
- Sepal Length, Sepal Width
- Sepal Width, Petal Length



Finally, I would conclude that the best combination of predictors are Petal Length & Petal Width with 3 centers. And the compactness of this model is 94.8% for training set 95% for testing set.

3.2 My R code is:

```
library(caret)
data <- read.delim(file= "data 4.2/iris.txt", header = TRUE, sep = "", dec = ".")
# split train and test sets, 8:2
datasplit <- createDataPartition(data[,5], 1, p=0.8, list=FALSE)
training_dataset  <- data[datasplit, ]
testing_dataset <- data[-datasplit, ]

#rep(0, nc-1)
elbowplot <- function(data1, nc, seed){
  tss <- (nrow(data1)-1)*sum(var(data1)*var(data1))
  for (i in 1:nc){
    set.seed(seed)
    tss[i] <- sum(kmeans(data1, centers=i, nstart=20)$withinss)}
  plot(1:nc, tss, type="b", xlab="Number of groups",
       ylab="Sum of squares within a group")}

elbowplot(training_dataset[,1,4], nc = 20, seed=5)


set.seed(5)
clustering_training <- kmeans(training_dataset[,1,4], centers = 4, nstart = 20)
clustering_training

set.seed(5)
clustering_testing <- kmeans(testing_dataset[,1,4], centers = 4, nstart = 20)
clustering_testing
```