3.1a

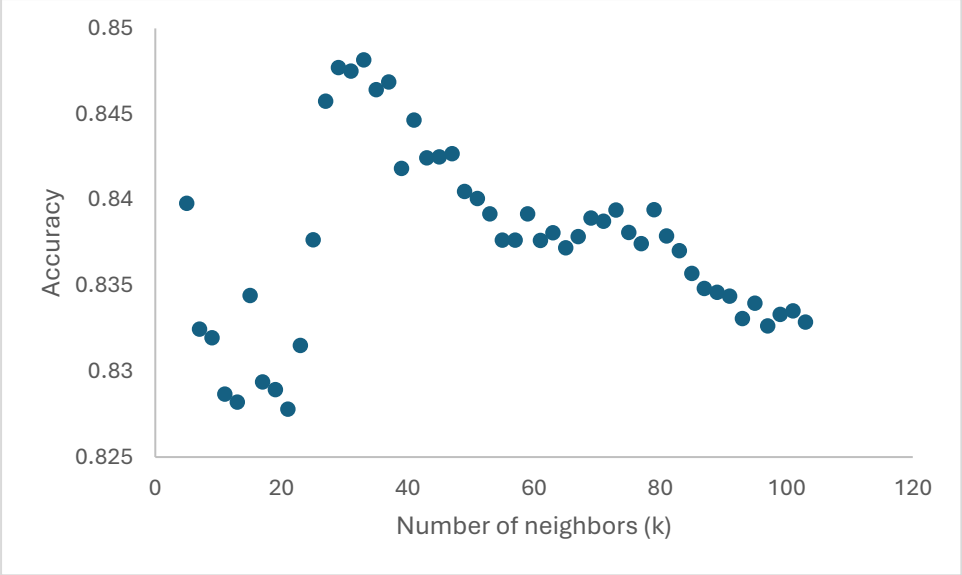
```
> data <- read.table("credit_card_data-headers.txt", stringsAsFactors = FALSE, header=TRUE)
> head(data)
  A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
1  1 30.83 0.000 1.25  1   0   1   1 202   0  1
2  0 58.67 4.460 3.04  1   0   6   1  43 560  1
3  0 24.50 0.500 1.50  1   1   0   1 280 824  1
4  1 27.83 1.540 3.75  1   0   5   0 100   3  1
5  1 20.17 5.625 1.71  1   1   0   1 120   0  1
6  1 32.08 4.000 2.50  1   1   0   0 360   0  1
```

The logic in my code is explained below:

1. There are several of types for cross-validation and one the most common function is caret package.
2. Split the data into training and testing sets (70% training, 30% testing)
3. Choosing the K-neighbors value for the testing: Small k value, the model is very sensitive to noise because it only looks at the small number neighbors. Large k value, the model becomes smoother and less sensitive to noise. Therefore, I chose maximum of k=50 for testing. This means the model will pick 50 values (5 to 103 with spacing = 2) as number of neighbors to perform Accuracy calculation.

```
knn_test <- train(as.factor(R1)~A1+A2+A3+A8+A9+A10+A11+A12+A14+A15,
                  train_data,
                  method = "knn",
                  trControl=trainControl(
                    method="repeatedcv",
                    number=k_cvtimes,
                    repeats=10),
```

4. The number of folds to use in k-fold cross-validation is an important hyperparameter that can affect the model's evaluation. This is number of cross validation the model will perform. The most common used values are K=5 or K=10. I set the K=10
5. Now, I use the training set with repeated-cross-validation method to fit the data to the K- nearest neighbors. The set.seed (42) to check the stability of the results
6. Using the train function from caret set, my input is train formula, train data, method, number of cross validation, repeats. The result is each k with corresponding accuracy
7. **I can see that when K=29, the accuracy is highest at 0.8456995~ 84.57%**



Then I use the knn trainning set to predict the results and accuracy of the validation set . I set up confusion matrix to evaluate the performance of a classification model by comparing predicted values against true values R1.

```
          Reference
Prediction   0   1
         0 100  24
         1  12  61
```

=>> Accuracy : 0.8173 ~ **81.73% = (100+61)/(100+61+24+12)**

In conclusion, after training the knn model on the training set, we observed a slight decrease in accuracy when the model was evaluated on the validation set. 84.57% accuracy for training set and 81.73% accuracy for validation set

3.1b

Split the data into three parts: 70% train, 15% test and 15% validation. The training data is used to fit the model.

Set the k max= 100 this time. For k in 1:k max, I use training data to fit the model and then using the model on the validation data to get prediction. After getting predictions, I calculate the accuracy of each k like in 3.1a.

```
knn.model <- kknn(R1~.,train,validation ,k=k,scale=TRUE)
```

```
for (k in 1:kmax) {
  knn.model <- kknn(R1~.,train,validation ,k=k,scale=TRUE)
  pred <- as.integer(fitted(knn.model)+0.5)
  accuracy[k] = sum(pred == validation$R1) / nrow(validation)
}
```

The best KKNN Model training accuracy is: 88.8% when k =7 on validation data.

Finally, we performed the kknn model on the test data and see the accuracy is 82%

```
knn_test <- kknn(R1~.,train,test,k=which.max(accuracy),scale=TRUE)
```

Using confusion matrix like in the previous question, I summarized the number of correct/incorrect prediction:

```
          Reference
'rediction  0  1
         0 45  5
         1 12 37

               Accuracy : 0.8283
```

The best KKNN Model training accuracy is: 82.8% when k =7 on test data.

4.1

My cousin have just recently opened a restaurant. In a restaurant setting, clustering can be an incredibly useful tool to improve customer service, enhance the dining experience, and boost profitability. Here are five potential predictors you might use to cluster customers in a restaurant setting:

-Menu

-Frequency of Visits

-Cutomer Feedback

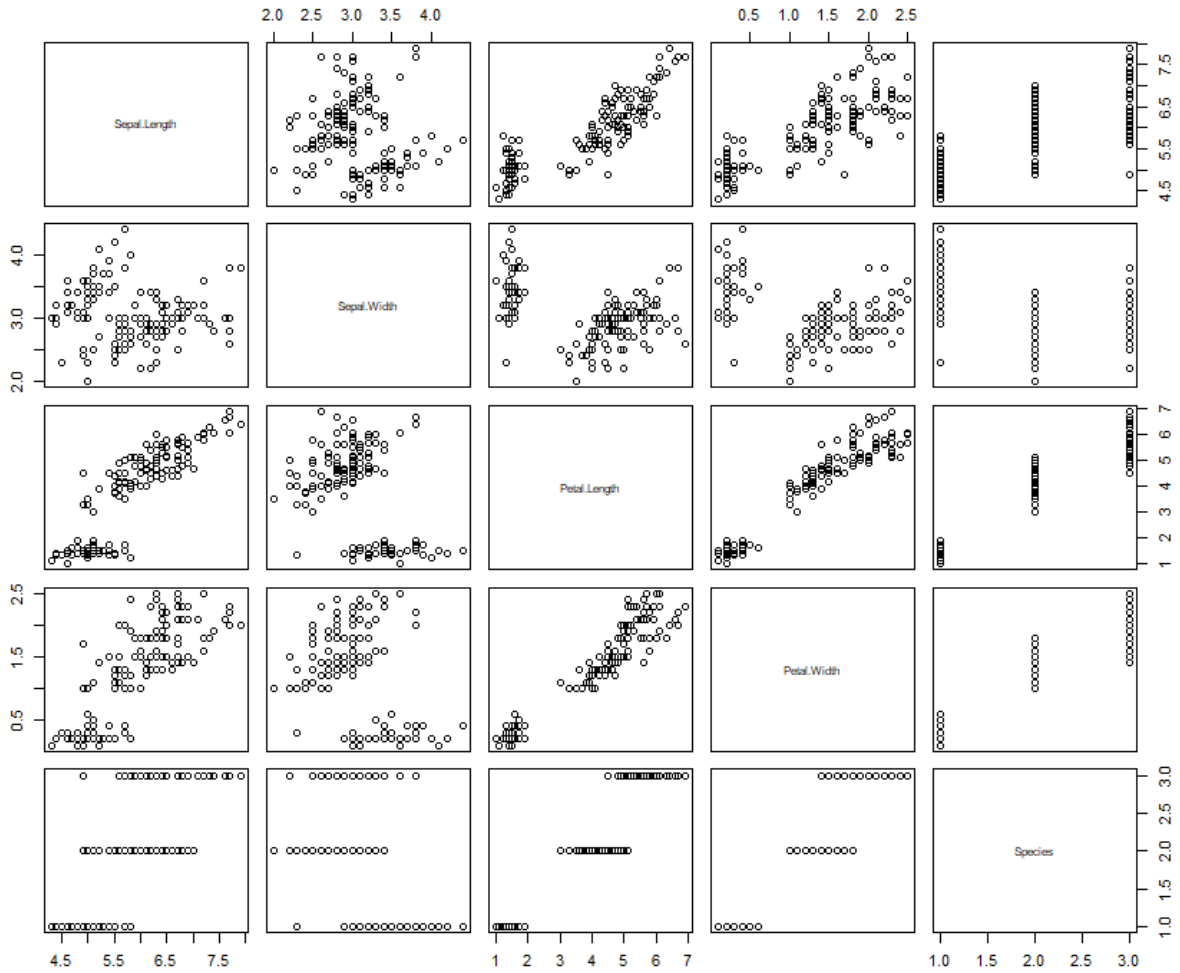
-Dish Frequency

-Customers Group (age, location,etc)

4.2

```
> iris <- read.table("iris.txt", header = TRUE)
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

This is the data before clustering:

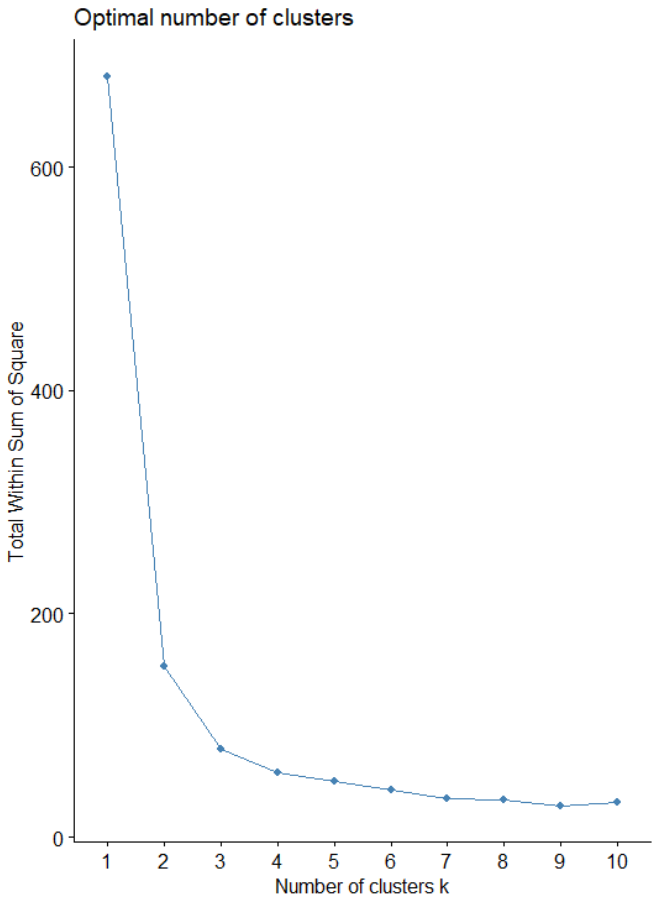


Using the code below, a plot of k cluster and accuracy method to determine the optimal number of clusters (k):

```
fviz_nbclust(scaled.data, kmeans, method = "wss")
```

This code determines the optimal number of clusters for k-means clustering using the Elbow Method based on the within-cluster sum of squares (WSS). WSS decreases as the number of clusters increases because adding more clusters

reduces the distance between data points and their cluster centroids. The Elbow method shows that the complexity of the model increases after k>3 without much improvement in separation of data points.



The optimal number of clusters is 3 based on this plot

When 3 clusters are used, the following partition is obtained:

	setosa	versicolor	virginica
1	0	39	14
2	50	0	0
3	0	11	36

The accuracy when all 4 predictors are used with 3 clusters were calculated

(50 + 39 +36)/ 150 = 83.33%. This accuracy can be further optimized by checking different combinations of predictors.