A0209349X Chong Woon Kiat

ST5227 Applied Data Mining Final Assessment

1. Classification and Regression Tree (CART) and K-Nearest Neighbor (KNN) method are selected to solve a classification problem.

2. Introduction to Classification and Regression Tree (CART):

    a. Classification and Regression Tree is also known as decision tree which has been widely used for classification or regression predictive modeling through a process of recursively splitting the data space.

    b. The idea is to divide predictor space set of possible feature variables into distinct and non-overlapping regions. For every observation in a region, prediction is made by taking mean of response in that region.

    c. Mathematically, sample space can be assumed into different region N_1, …, N_H region, and the expected value of an observation with predictors x_1, …, x_p can be approximated by function

    $$E(Y) = m(x_1, \dots, x_p) = \sum_{h=1}^{H} c_h I(x \in N_h)$$

    Where I is an indicator function and c_h is estimated the average value of the response y of sample $S_0 = \{(x_{i,1}, \dots, x_{i,p}, y_i) : i = 1, \dots, n\}$ found in the region N:

    $$\hat{c}_h = \frac{\sum_{i=1}^{n} y_i I(x_i \in N_h)}{\sum_{i=1}^{n} I(x_i \in N_h)}$$

    d. The choose the optimal x_k' to partition each variable x_k, we minimize the function

    $$Q_k = \min_{c_1, c_2, x_k} \sum_{x_{i,k} \le x_k} (Y_i - c_1)^2 + \sum_{x_{i,k} > x_k} (Y_i - c_2)^2$$

    Leave-one-out cross validation method can be used to find x_k' for each x_k. Cross validation values of all variables are compared, the one with the smallest value will be the first splitting variable and node. The same procedure is applied to the next splitting until no more splitting is necessary.

3. Introduction to K-Nearest Neighbor (KNN):

    a. K-Nearest Neighbor has been widely used for both classification and regression problem. It assumes similar things exist in close proximity.

b. The idea is to locate the k nearest neighbors of a test sample and this test sample is classified by a majority vote.

c. Mathematically, the expected value of an observation is estimated by taking average of k-nearest observations, and can be approximated by function

$$E(Y) = m(X) = \frac{\sum_{i=1}^{n} y_i I(X_i \in D_x)}{\sum_{i=1}^{n} I(X_i \in D_x)}$$

Where $D_x = \{ X_i : X_i \; is \; one \; one \; of \; the \; k \; nearest \; observations \; to \; X \}$

Nearest neighbors are determined based on distance function. Typical distance functions are:

- Euclidean distance: $\{\sum_{s=1}^{p}(x_{is} - x_{js})^2\}^{1/2}$
- Absolute distance: $\sum_{s=1}^{p} |x_{is} - x_{js}|$
- Minkowski distance: $\{\sum_{s=1}^{p}(x_{is} - x_{js})^q\}^{1/q}$

d. Similarly, optimal k is determined through leave-one-out cross validation

$$\hat{y}_i(k) = \frac{\sum_{j=1, j\neq i}^{n} y_i I\{X_j \in D_x(X_i)\}}{\sum_{j=1, j\neq i}^{n} I\{X_j \in D_x(X_i)\}} \; , k = 1, 2, \dots, n$$

With CV values

$$CV(k) = \frac{1}{n}\sum_{i=1}^{n}\{y_i - \hat{y}_i(k)\}^2$$

K is selected by

$$\arg\min_k CV(k)$$

4. Dataset “TitanicSurvival” built-in in R is chosen and a classification model is built to predict survival of a passenger. The data consists of 1046 observations and 4 columns:
    - Survived: Categorical data with value “yes”, “no”
    - Sex: Categorical data with value “male”, “female”
    - Age: Continuous data
    - PassengerClass: Categorical data with vale “1st”, “2nd”, “3rd”

```
> data("TitanicSurvival")
> head(TitanicSurvival,3)
                          survived    sex     age passengerClass
Allen, Miss. Elisabeth Walton      yes female 29.0000            1st
Allison, Master. Hudson Trevor     yes   male  0.9167            1st
Allison, Miss. Helen Loraine        no female  2.0000            1st
> df <- na.omit(TitanicSurvival)
```

```
> nrow(df)
[1] 1046
```

a. Factorize categorical columns

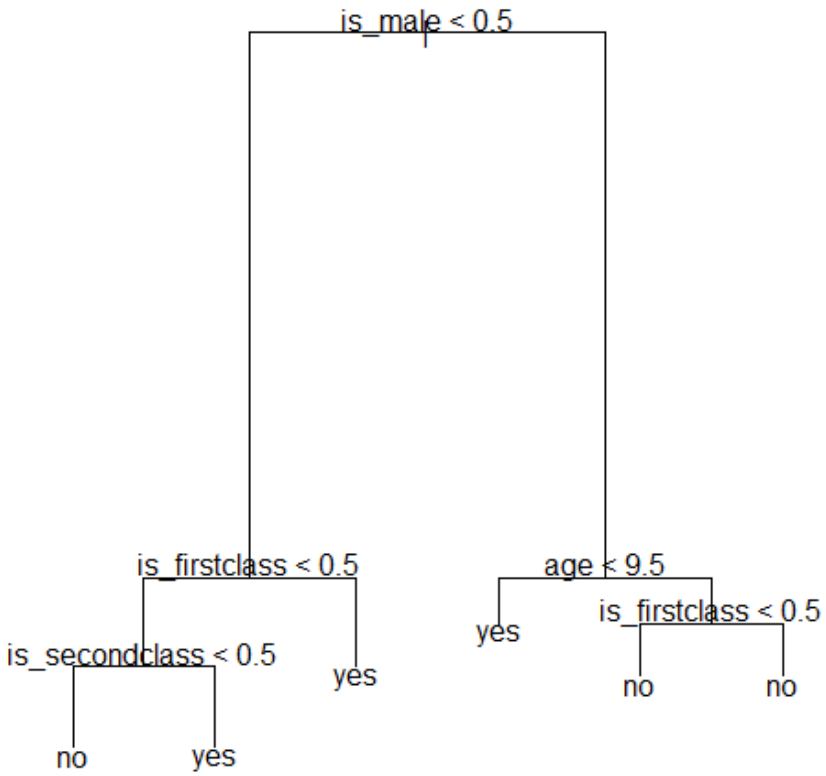
```
> df$is_male = (df$sex == "male") + 0
> df$is_firstclass = (df$passengerClass == "1st") + 0
> df$is_secondclass = (df$passengerClass == "2nd") + 0
```

b. Split 80% data as train dataset, remaining 20% as test dataset. Keep only columns of interest (“survived” as target variable; “age”, “is_male”, “is_firstclass”, “is_secondclass” as predictors)

```
> test_index = seq(1,1045,5)
> df2 = df[,c("survived","age","is_male","is_firstclass","is_secondclass
")]
```

5. CART model
   a. Build a CART model with train dataset. From the summary, it is observed than the error rate is 22%

```
> library(tree)
> modelcart = tree(survived~. , df2[-test_index,])
> plot(modelcart)
> text(modelcart)
```



```
> modelcart
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 837 1132.00 no ( 0.59140 0.40860 )
   2) is_male < 0.5 308  341.90 yes ( 0.24351 0.75649 )
     4) is_firstclass < 0.5 202  260.70 yes ( 0.34653 0.65347 )
       8) is_secondclass < 0.5 124  171.90 no ( 0.50000 0.50000 ) *
       9) is_secondclass > 0.5 78   51.59 yes ( 0.10256 0.89744 ) *
     5) is_firstclass > 0.5 106   40.30 yes ( 0.04717 0.95283 ) *
   3) is_male > 0.5 529  538.20 no ( 0.79395 0.20605 )
     6) age < 9.5 32   43.86 yes ( 0.43750 0.56250 ) *
     7) age > 9.5 497  473.20 no ( 0.81690 0.18310 )
      14) is_firstclass < 0.5 377  295.10 no ( 0.86737 0.13263 ) *
      15) is_firstclass > 0.5 120  154.10 no ( 0.65833 0.34167 ) *


> summary(modelcart)

Classification tree:
tree(formula = survived ~ ., data = df2[-test_index, ])
Number of terminal nodes:  6
Residual mean deviance:  0.9108 = 756.8 / 831
Misclassification error rate: 0.2151 = 180 / 837
```

b. From the tree, we can tell
   - a female with first class or second class is more likely to survive than a female with third class.
   - A male kid with age lower than 9.5 is likely to survive too.

c. Predict survival of passenger in the test dataset, and calculate the error rate. Obtained error rate of 21% in test dataset which is similar to the error obtained from train dataset, suggesting that the model is not overfitting.

```
> predcart = predict(modelcart, df2[test_index,], type="class")
> errorcart = 1 - sum((df2[test_index,"survived"]==predcart) + 0)/length(d
f2[test_index,"survived"])
> errorcart
[1] 0.2105263
```

6. KNN model
   a. Check_error(k) function is to return the CV value of a k value by leaving one observation out every iteration from the train dataset.
   b. Distance = 2 indicates that Euclidean distance is used.
   c. Gaussian kernel is applied as weight to the k neighbours.

```
> library(kknn)
>
> check_error = function(k) {
+
+   errork <- rep(0,(nrow(df2[-test_index,])))
+
+   for (i in 1:(nrow(df2[-test_index,]))){
+
+     modelknn = kknn(survived~., df2[-test_index,][-i,], df2[-test_inde
x,][i,], kernel="gaussian", distance=2, k = k)

+     predknn = predict(modelknn)
+     errork[i] = (df2[-test_index,"survived"][i] != predknn) + 0
```

```
+    }
+
+    meanerrork = mean(errork)
+    return(meanerrork)
+ }
```

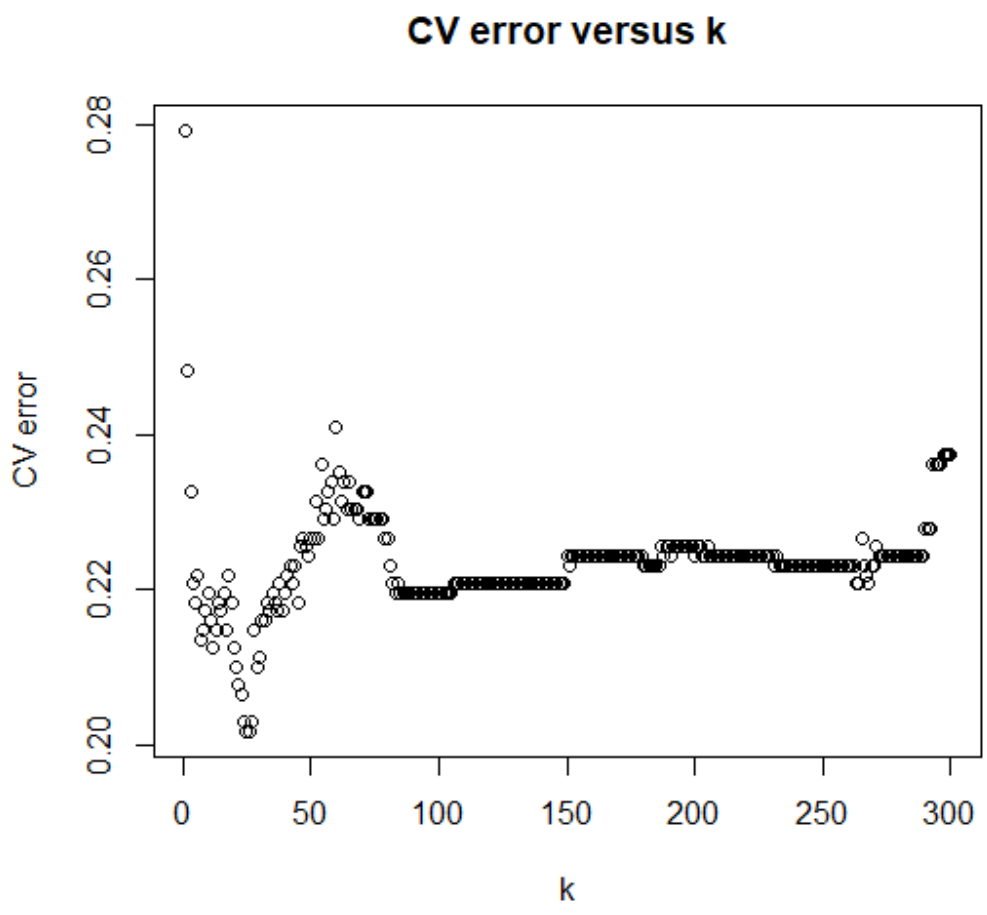
d. Use the check_error(k) function to loop through k from 1 to 300 to find the k with lowest cross validation value. K = 25 is selected with CV error at 20%.

```
> er = rep(0,50)
> for (k in 1:50){
+    er[k] = check_error(k)
+ }

> plot(er, main = "CV error versus k", ylab = "CV error", xlab = "k" )


> min(er)
[1] 0.2016706
> which(er == min(er))
[1] 25 26

> best_k = which(er == min(er))[1]
> best_k
[1] 25
```



e. Fit the model with test dataset to check error. Error of 17% is obtained, indicating that the model is not overfitting.

```
> modelknn = kknn(survived~., df2[-test_index,], df2[test_index,], kernel=
"gaussian", distance=2, k = best_k)
> predknn = predict(modelknn)
> errorknn = 1 - sum((df2[test_index,"survived"]==predknn) + 0)/length(df2
[test_index,"survived"])
> errorknn
[1] 0.1722488
```

7. Comparing the results oh CART and KNN method, KNN has a better performance with lower error in test dataset:

Method	Train Error	Test Error
CART	22%	21%
KNN	20%	17%

8. One thing to note is that KNN is computationally more expensive and it takes up large memory storing and looking up from the entire training set while doing prediction/ CART is relatively faster in computation due to absence of calculation in its classification process.