Practice of Classifications

Data set: https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation

Requirement
Build multiple models by using KNN, Naïve Bayes, Logistic regression, Decision Tree classification algorithms
Evaluate them and find which model is the best
Upload your R codes and outputs step by step, along with appropriate/necessary explanations

The student who got the best accuracy will get a bonus point (+10)
Hint: there are several parameters in each algorithm, you can tune up them in the R environment
Hint: this is a multi-class classification task

- Hints

Make a decision about features and labels
Make a decision about evaluations (strategy and metrics)
Make a decision about which algorithms to be used
Preprocess your data according to the requirements in each algorithm
Run classifications on the preprocessed data

1. Whether of not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker
2. Course instructor (categorical, 25 categories)
3. Course (categorical, 26 categories)
4. Summer or regular semester (binary) 1=Summer, 2=Regular
5. Class size (numerical)
6. Class attribute (categorical) 1=Low, 2=Medium, 3=High

Step 1: Understand the attributes

Attribute Information:
1. Whether or not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker => **binary but should be treated as nominal**
2. Course instructor (categorical, 25 categories) => **int but should be treated as nominal**
3. Course (categorical, 26 categories) => **int but should be treated as nominal**
4. Summer or regular semester (binary) 1=Summer, 2=Regular => **binary but needs to be treated as nominal**
5. Class size (numerical)
6. Class attribute (categorical) 1=Low, 2=Medium, 3=High => **int but should be treated as nominal**

Step 2: Loading data

```
> mydata = read.table("tae.csv",sep=",")
> head(mydata)
  V1 V2 V3 V4 V5 V6
1  1 23  3  1 19  3
2  2 15  3  1 17  3
3  1 23  3  2 49  3
4  1  5  2  2 33  3
5  2  7 11  2 55  3
6  2 23  3  1 20  3
> str(mydata)
'data.frame':   151 obs. of  6 variables:
 $ V1: int  1 2 1 1 2 2 2 2 1 2 ...
 $ V2: int  23 15 23 5 7 23 9 10 22 15 ...
 $ V3: int  3 3 3 2 11 3 5 3 3 3 ...
 $ V4: int  1 1 2 2 2 1 2 2 1 1 ...
 $ V5: int  19 17 49 33 55 20 19 27 58 20 ...
 $ V6: int  3 3 3 3 3 3 3 3 3 3 ...
```

Note all variables are loaded as integers, which are not right
We need to convert them to the right variable types first.

```
> mydata$V1=factor(mydata$V1)
> mydata$V2=factor(mydata$V2)
> mydata$V3=factor(mydata$V3)
> mydata$V4=factor(mydata$V4)
> mydata$V6=factor(mydata$V6)
> str(mydata)
'data.frame':   151 obs. of  6 variables:
 $ V1: Factor w/ 2 levels "1","2": 1 2 1 1 2 2 2 2 1 2 ...
 $ V2: Factor w/ 25 levels "1","2","3","4",..: 23 15 23 5 7 23 9 10 22 15 ...
 $ V3: Factor w/ 26 levels "1","2","3","4",..: 3 3 3 2 11 3 5 3 3 3 ...
 $ V4: Factor w/ 2 levels "1","2": 1 1 2 2 2 1 2 2 1 1 ...
 $ V5: int  19 17 49 33 55 20 19 27 58 20 ...
 $ V6: Factor w/ 3 levels "1","2","3": 3 3 3 3 3 3 3 3 3 3 ...
```

STEP 3: Is there any missing value in dataset?

```
> sum(is.na(mydata[c("V1")]))
[1] 0
> sum(is.na(mydata[c("V2")]))
[1] 0
> sum(is.na(mydata[c("V3")]))
[1] 0
> sum(is.na(mydata[c("V4")]))
[1] 0
> sum(is.na(mydata[c("V5")]))
[1] 0
> sum(is.na(mydata[c("V6")]))
[1] 0
```

There is no missing value in dataset!

Step 4: Building models:

I am going to use four algorithms to build models.

- Naïve Bayes ➔ features must be categorical variables
- Logistic Regression ➔ features could be any type of the variables
- K-Nearest Neighbor ➔ features must be normalized numerical variables
- Decision Tree ➔ features could be any data type

Since, the size of data is small, for all models the evaluation method will be N-Fold cross validation.

Step 4.1. Naïve Bayes Model:

```
install.packages('naivebayes', dependencies = TRUE)
library(naivebayes)
install.packages('caret', dependencies = TRUE)
library(caret)

#Naive Bayes Model
#Data preprocessing
data_NB = mydata
data_NB$V1=factor(data_NB$V1)
data_NB$V2=factor(data_NB$V2)
data_NB$V3=factor(data_NB$V3)
data_NB$V4=factor(data_NB$V4)
data_NB$V5=factor(cut(data_NB$V5,3))
data_NB$V6=factor(data_NB$V6)

x_NB = mydata[,-6]
head(x_NB)

y_NB = mydata[,6]
head(y_NB)
```

Define the 10-fold cross validation and run it.

```
NBmodel = train(x_NB,y_NB, method = 'nb',trControl=trainControl(method='cv',number=10),
na.action=na.pass)
print(NBmodel)
> print(NBmodel)
Naive Bayes

151 samples
  5 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 136, 136, 135, 136, 136, 136, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE      0.5171429  0.2735989
   TRUE      0.5175595  0.2737973

Tuning parameter 'fL' was held constant at a value of 0
Tuning parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust = 1.
```

The classification accuracy by Naïve Bayes on 10-fold cross validation is 51.71%

Step 4.2. Logistic Regression Model:

```
########## Logistic Regression ###########

head(mydata)
# extract features
x_LR = mydata[,-6]
head(x_LR)
install.packages("dummies")
library("dummies")
# convert features to dummy variables
df_LR = dummy.data.frame(x_LR,names=c("V1","V2","V3","V4"))
summary(df_LR)
# extract labels
y_LR=mydata$V6
# build models
LRmodel = train(df_LR,y_LR, method = 'multinom',trControl=trainControl(method='cv',
number=10),na.action=na.pass)
print(LRmodel)
> print(LRmodel)
Penalized Multinomial Regression

151 samples
 56 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 136, 135, 136, 136, 136, 136, ...
Resampling results across tuning parameters:

  decay  Accuracy   Kappa
  0e+00  0.5234524  0.2847423
  1e-04  0.5367857  0.3047423
  1e-01  0.5104762  0.2646674

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was decay = 1e-04.
```

Accuracy for Logistic Regression model is 53.68%.

Step 4.3. Decision Trees Model:

```
install.packages("rpart")
library(rpart)

fit <- train(V6~., data = mydata, method = "rpart",
             trControl=trainControl(method = "cv", number = 10),
             tuneLength = 10,
             parms=list(split='information'))

print(fit)
```

Step 4.4. KNN Model:

```
########## KNN ###########
x_knn = mydata[,-6]
head(x_knn)

#Data Preprocessing
#Coverting nominal variables to dummy
x_knn = dummy.data.frame(x_knn,names=c("V1","V2","V3","V4"))
head(x_knn)
#Applying min-max normalization
x_knn = as.data.frame(apply(x_knn,2, FUN = function(x) (x - min(x))/(max(x)-min(x))))
head(x_knn)
# build models
KNN = train(x_knn,mydata[,6],'knn',trControl=trainControl(method='cv',number=10),
tuneGrid= expand.grid(k = 1:10))
print(KNN)
```

```
> print(KNN)
k-Nearest Neighbors

151 samples
 56 predictor
  3 classes: '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 136, 136, 135, 136, 136, 136, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.6447619  0.46654240
   2  0.4401190  0.15874903
   3  0.3928571  0.08909125
   4  0.4182738  0.12399103
   5  0.4450595  0.16597951
   6  0.4379167  0.15745426
   7  0.4561310  0.18243801
   8  0.4436905  0.16523542
   9  0.4574405  0.18540052
  10  0.4578571  0.18723366

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 1.
```

When k=1, we got the best accuracy: 64.47%