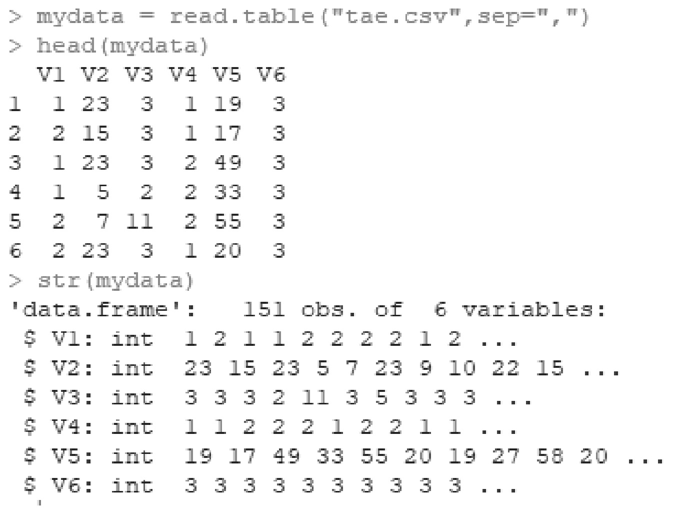
Step 1: Understand the attributes

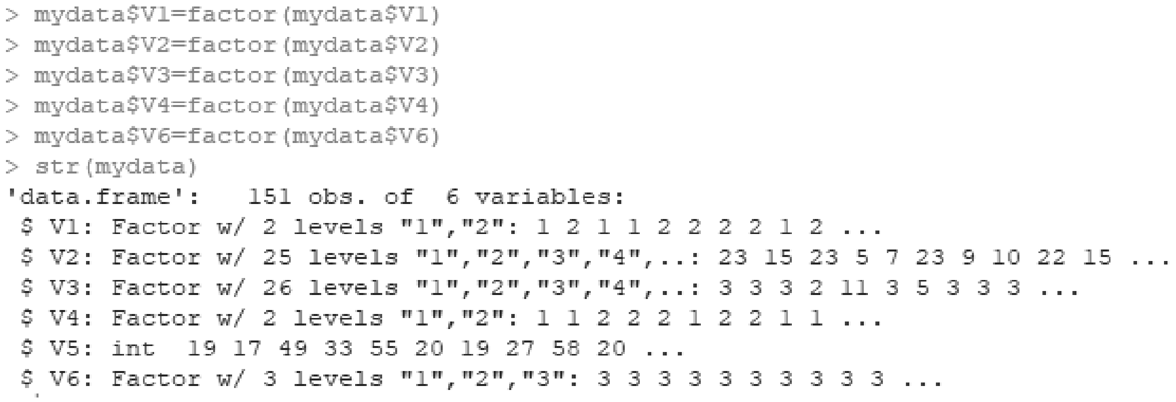
Attribute Information:
1. Whether or not the TA is a native English speaker (binary); 1=English speaker, 2=non-English speaker => **binary but should be treated as nominal**
2. Course instructor (categorical, 25 categories) => **int but should be treated as nominal**
3. Course (categorical, 26 categories) => **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



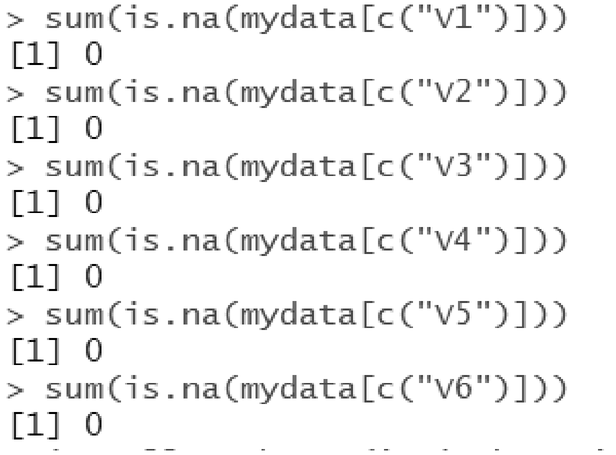
```
> mydata = read.table("tae.csv",sep=",")
> head(mydata)
  V1 V2 V3 V4 V5 V6
1  1 23  3  1 19  3
2  2 15  3  1 17  3
3  1 23  3  2 49  3
4  1  5  2  2 33  3
5  2  7 11  2 55  3
6  2 23  3  1 20  3
> str(mydata)
'data.frame':   151 obs. of  6 variables:
 $ V1: int  1 2 1 1 2 2 2 2 1 2 ...
 $ V2: int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

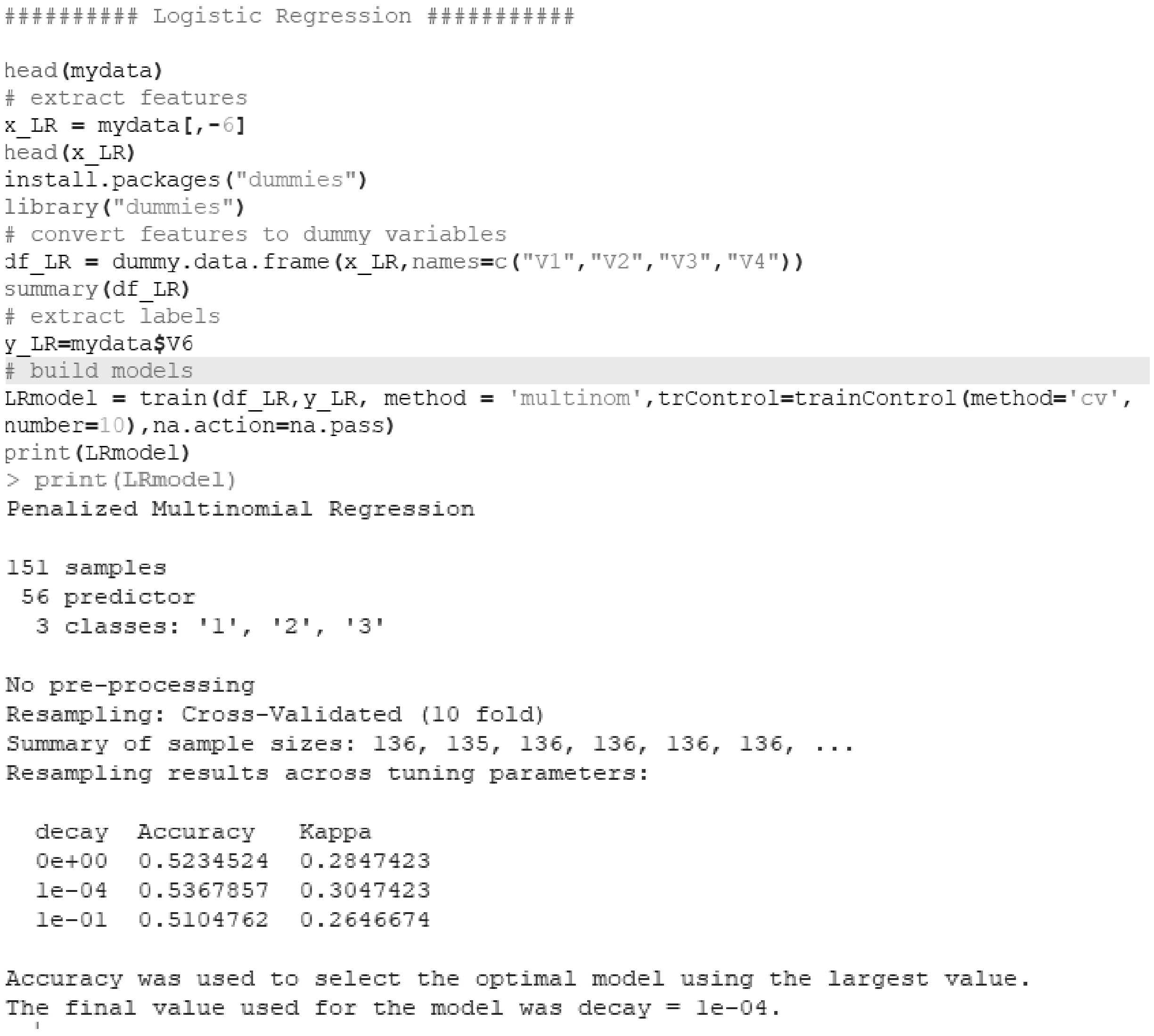
Step 4: Building models:

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 4.2. 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%.