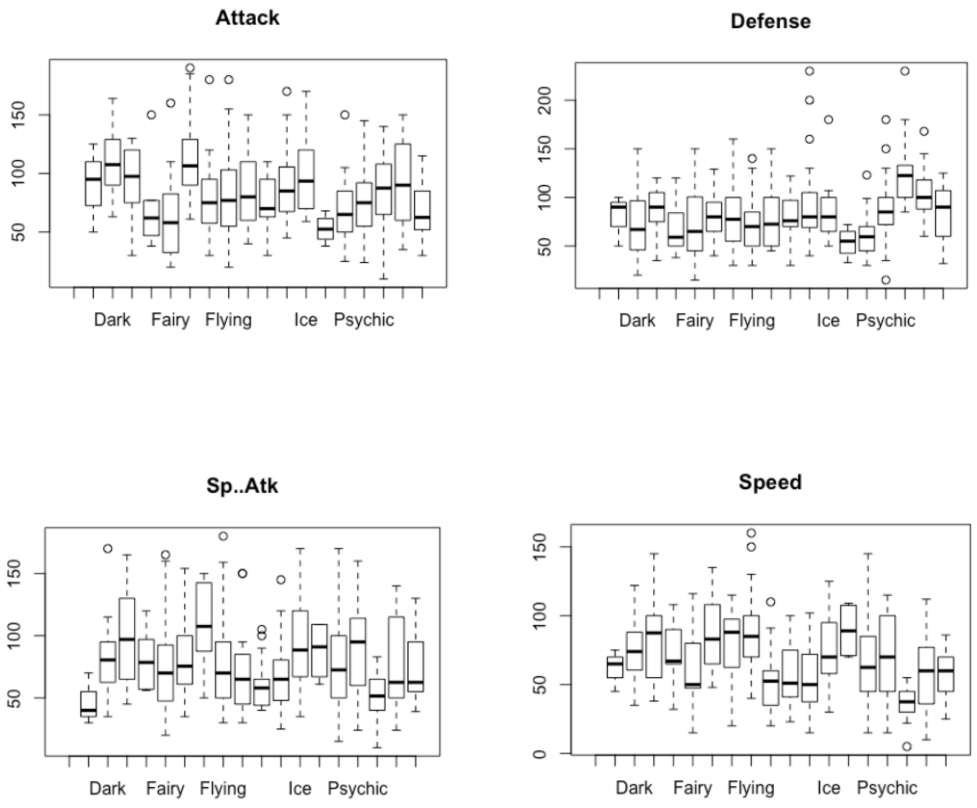
STSCI-4740

Machine Learning - Pokemon Project

Pretish Kuruvila and Will Kretz

Question 1:

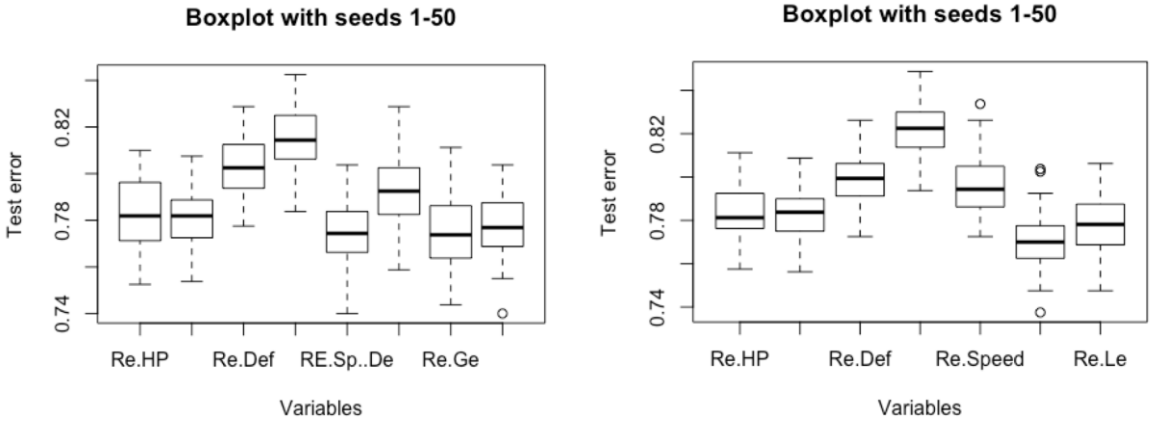
We plotted boxplots for all the predictors to show the variation for the predictors. From the plot we can see the predictors: HP, Total, Attack, Defense, Sp..Atk, Speed have significant variation, so we initially choose HP, Total, Attack, Defense, Sp..Atk, Speed. And we deleted the predictor Total due to the collinearity.

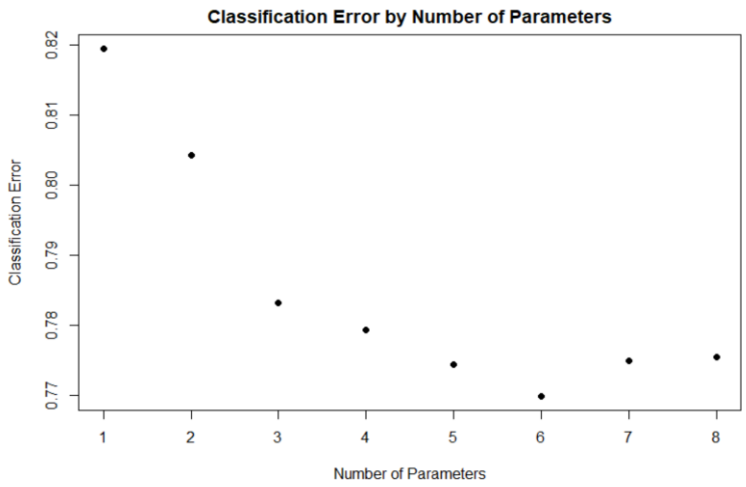


In order to predict the type of the various pokemon, we employ linear discriminant analysis to fit the model to the categorical data. With 18 response values and only 8

parameters, it seems unlikely that the model will be capable of predicting the large response set. The variables had to first be tested so that we knew which ones to use. Thus, backward selection was used to determine which variables to consider when modeling the system. To improve the sampling of our dataset, k fold cross validation was used. Because we wanted to eliminate any errors associated with sampling, each model was run over 50 random seeds, and the average classification error rate was found. The model that produced the smallest classification error was then used for the subsequent step in backward selection. In other if a k-variable model with a certain variable removed produced the lowest classification error of all the k-variable models, the model with that variable removed was then used for the next step of backward selection. In the end, the order of removal was found to be Sp. Def, Generation, Legendary, HP, Speed, Defense, and finally Attack. All that remained was the variable Sp. Attack.

With the 8 models, we then compared their test errors and chose the one that produced the lowest classification error. This results of the 8 models are shown below along with box plots that show the first two steps of the backward selection process:





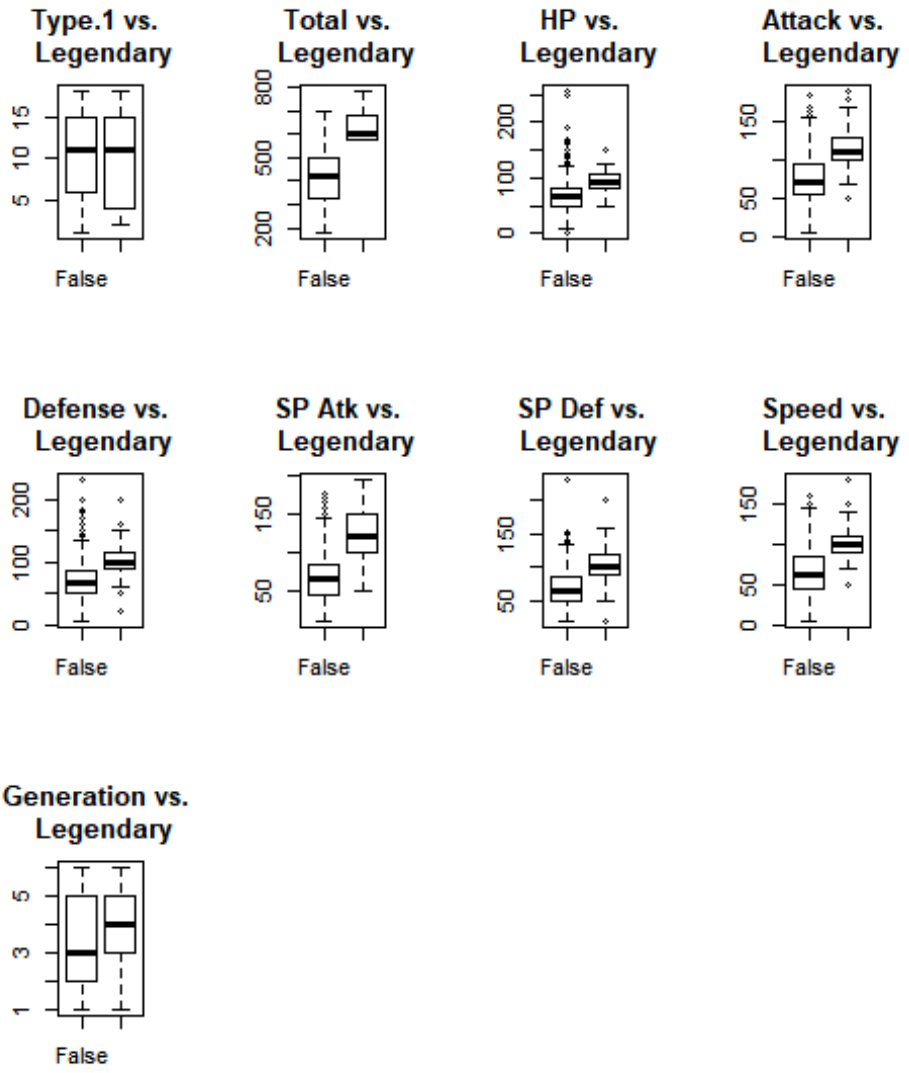
The model containing 6 parameters, Sp. Attack, Attack, Defense, Speed, HP, and Legendary, predicted the Type1 of each Pokemon with the lowest classification error. This model predicts Type 1 with a classification error of 76.985%, which is extremely large. Thus, it is not possible to predict Type1 of the Pokemon with linear discriminant analysis.

We then used KNN, decision tree, and random forest to predict the Type1 of the Pokemon with the hope a different model would produced better results. The results are shown in a table below. Clearly we never got good results. Thus we conclude that you cannot predict the Pokemon Type with a reasonable classification error using the given parameters. We used the same methods of prediction for Type.2, results are shown below in the summary table. It was again found that we could not predict the type with a reasonable classification error rate.

Classification Error Rates of the Different Methods (%)

Method	Type 1	Type 2
Linear Discriminant	77%	76%
KNN	78%	56%
Decision Tree	90%	54%
Random Forest	74%	57%

Question 2:



From the boxplots above, the features that seem most likely to be useful in predicting Legendary are Attack, Defense, SP Atk, SP Def and Speed. These variables show a clear variability between the two variables False and True. Additionally, Total is not considered a possible predictor because it is a linear combination of all other variables.

The results from the code (see Appendix 1) are as follows:

```
## The test error under logistic regression is:  0.0675
## The test error under QDA is:  0.0725
```

```
## The 10-fold cross validation error under QDA is:  0.065
## The 10-fold cross validation error under logistic regression is:  0.0575
## The test error for KNN 10-fold cross validation is: 0.05875
## The test error for LDA 10-fold cross validation is: 0.06625
## The test error for LDA validation set is: 0.07
## The test error for KNN validation set is: 0.0575
```

Summary Table

Method	Validation Set Error	10 Fold Cross Validation (CV) Error
Logistic Regression	0.0675	0.05750
Linear Discriminant Analysis (LDA)	0.0700	0.06625
Quadratic Discriminant Analysis (QDA)	0.0725	0.06500
K-Nearest Neighbours (KNN)	0.0575	0.05875

To begin with, we split the data (n=800) into half to obtain the training and testing data sets. Then, we fit the Logistic Regression, LDA, QDA and KNN models on the training data set. The test errors for each model is obtained through running the models on the test data set. Of the four errors listed above in the table, KNN has the smallest validation set test error.

In order to obtain the 10-fold cross validation errors, we created the function for the 4 methods. Among all the CV errors generated, Logistic Regression has the lowest error rate.

However, under comparison of the test errors between Logistic Regression and KNN, it is noticed that KNN has the lower error when considering both Validation set and 10-fold CV together.

Therefore, KNN will give the more accurate result in predicting whether a Pokemon is legendary or not with predictors Attack, Defense, SP Atk, SP Def and Speed.

Appendix 1 – R Code

```
##############################################
```

```
################ QUESTION 1 ################
############################################




############################################
################ QUESTION 2 ################
############################################

# Load in the data set
pokemon<-read.csv('C:/Users/yunat/Desktop/STSCI
4740/PROJECT/Pokemon.csv', header = TRUE)

# Remove ID column
pokemon <- pokemon[, -1]

# Code index for True and False where Legendary = 1 when true
pokemon$isLegendary <- 1
pokemon$isLegendary[pokemon$Legendary == "False"] <- 0
pokemon$isLegendary <- as.factor(pokemon$isLegendary)

pokemon$Type1 <- as.numeric(pokemon$Type.1)

###########
# Boxplot #
###########

# Identify possible predictors using boxplot
```

```
par(mfrow=c(2,4))

boxplot(Type1~Legendary, data=pokemon, main="Type.1 vs. \n Legendary")

boxplot(Total~Legendary, data=pokemon, main="Total vs. \n Legendary")

boxplot(HP~Legendary, data=pokemon, main="HP vs. \n Legendary")

boxplot(Attack~Legendary, data=pokemon, main="Attack vs. \n Legendary")

boxplot(Defense~Legendary, data=pokemon, main="Defense vs. \n Legendary")

boxplot(Sp..Atk~Legendary, data=pokemon, main="SP Atk vs. \n Legendary")

boxplot(Sp..Def~Legendary, data=pokemon, main="SP Def vs. \n Legendary")

boxplot(Speed~Legendary, data=pokemon, main="Speed vs. \n Legendary")

boxplot(Generation~Legendary, data=pokemon, main="Generation vs. \n Legendary")


#########################

# Splitting the Dataset #

#########################


# Split the data set into equal size training set and testing set

set.seed(1)


n=dim(pokemon)[1]

train <- sample(1:n, n/2)

pokemon_train<- pokemon[train,]

pokemon_test <- pokemon[-train,]
```

```
##########################################
# Validation Set Test Error for Logistic #
##########################################

# Fit logistic regression
lgr.fit1 <- glm(isLegendary ~ Attack + Defense + Sp..Atk + Sp..Def + Speed, data = pokemon_train, family=binomial)

# Test Error (Validation Set)
legendary_test <- pokemon$isLegendary[-train]
glm.probs=predict(lgr.fit1,pokemon_test,type="response")
glm.pred=rep("0", dim(pokemon)[1]/2)
glm.pred[glm.probs>.5]="1"
log_te <- mean(glm.pred != legendary_test)
cat("The test error under logistic regression is: ", log_te)

####################################
# Validation Set Test Error for QDA #
####################################

# Fit quadratic discriminant
library(MASS)
qda.fit <- qda(isLegendary ~ Attack + Defense + Sp..Atk + Sp..Def + Speed, data=pokemon_train)
```

```
# Test Errors (Validation Set)
legendary_test <- pokemon$isLegendary[-train]
qda.predict <- predict(qda.fit, pokemon_test)
qda.class <- qda.predict$class
qda_te <- mean(qda.predict$class != legendary_test)
cat("The test error under QDA is: ", qda_te)

############################
# 10-Fold CV Error for QDA #
############################

# K-fold for QDA
k_fold_xval <- function(K, data){
  n <- dim(data)[1]
  sample_vec <- c(1:n)
  fold_size <- n / K
  fold_indeces <- vector("list", K)
  for(i in 1:K){
    fold_index_test <- sample(sample_vec, fold_size)
    fold_indeces[[i]] <- fold_index_test
    sample_vec <- sample_vec[-fold_index_test]
  }
  cv_within_fold <- c(rep(0, K))
  for(j in 1:K){
    test_index <- fold_indeces[[j]]
    dat_test <- data[test_index, ]
    dat_train <- data[-test_index, ]
```

```
    qda_fit <- qda(isLegendary ~ Attack + Defense + Sp..Atk +
Sp..Def + Speed, data = dat_train)

    qda_predict <- predict(qda_fit, dat_test)


    cv_within_fold[j]      <-      mean(qda_predict$class      !=
dat_test$isLegendary)

  }

  mean(cv_within_fold)

}


set.seed(1)


kfold_er_qda <- k_fold_xval(K = 10, data = pokemon)

cat("The  10-fold  cross  validation  error  under  QDA  is:  ",
kfold_er_qda)


##############################

# 10-Fold CV Error Logistics #

##############################


# K-fold for logistics

k_fold_xval.log <- function(K, data){

  n <- dim(data)[1]

  sample_vec <- c(1:n)

  fold_size <- n / K

  fold_indeces <- vector("list", K)

  for(i in 1:K){

    fold_index_test <- sample(sample_vec, fold_size)
```

```
    fold_indeces[[i]] <- fold_index_test
    sample_vec <- sample_vec[-fold_index_test]
  }
  cv_within_fold <- c(rep(0, K))
  for(j in 1:K){
    test_index <- fold_indeces[[j]]
    dat_test <- data[test_index, ]
    dat_train <- data[-test_index, ]
    log.fit <- glm(isLegendary ~ Attack + Defense + Sp..Atk +
Sp..Def + Speed, data = dat_train, family = binomial)
    log_probs <- predict(log.fit, dat_test,type="response")
    log_predict<-c(rep(0, nrow(dat_test)))
    log_predict[log_probs>.5] <- 1

    cv_within_fold[j]        <-        mean(log_predict        !=
dat_test$isLegendary)
  }
  mean(cv_within_fold)
}

set.seed(1)
kfold_er_log <- k_fold_xval.log(K = 10, data = pokemon)
cat("The 10-fold cross validation error under logistic
regression is: ", kfold_er_log)

######################
# Function for CV KNN #
```

```
######################

Data=read.csv('Pokemon.csv',header=TRUE)  #Name  for  the  pokemon
data is 'Data'

n <- nrow(Data)


cv.knn <-

  function (Data, features, yname, K, seed) {

    set.seed(seed)

    Data$Legendary=as.character(Data$Legendary)

    Data$Legendary[Data$Legendary == "True"] <- 1

    Data$Legendary[Data$Legendary == "False"] <- 0

    library(MASS)

    library(class)

    #partition the data into K subsets

    f <- ceiling(n/K)

    s <- sample(rep(1:K, f), n)

    #generate indices 1:10 and sample n of them

    # K fold cross-validated error

    CV=NULL

    for (i in 1:K) { #i=1

      test.index <- seq_len(n)[(s == i)] #test data

      train.index <- seq_len(n)[(s != i)] #training data

      #observed test set y

      knn.y <- Data[test.index, yname]

      #predicted test set y

      knn.predy=knn(Data[train.index,features],
Data[test.index,features],Data[train.index, yname],k=10)

      #observed - predicted on test data
```

```
    error= mean(knn.y!=knn.predy)
    #error rates
    CV=c(CV,error)
  }
  #Output
  list( K = K,
        knn_error_rate = mean(CV), seed = seed)
}
er_knn=cv.knn(Data,features=c("Attack","Defense","Sp..Atk","Sp..
Def","Speed"), yname="Legendary", K=10, seed=1)
cat("The test error for KNN 10-fold cross validation
is:",er_knn$knn_error_rate)


# K Fold CV LDA
n <- nrow(Data)
library(MASS)
cv.lda <-
  function (Data, model, yname, K, seed) {

    set.seed(seed)
    Data$Legendary=as.character(Data$Legendary)
    Data$Legendary[Data$Legendary == "True"] <- 1
    Data$Legendary[Data$Legendary == "False"] <- 0

    #partition the data into K subsets
    f <- ceiling(n/K)
    s <- sample(rep(1:K, f), n)
    #generate indices 1:10 and sample n of them
```

```
    # K fold cross-validated error
    CV=NULL
    for (i in 1:K) { #i=1
      test.index <- seq_len(n)[(s == i)] #test data
      train.index <- seq_len(n)[(s != i)] #training data
      #model with training data
      lda.fit=lda(model, data=Data[train.index,])
      #observed test set y
      lda.y <- Data[test.index, yname]
      #predicted test set y
      lda.predy=predict(lda.fit, Data[test.index,])$class
      #observed - predicted on test data
      error= mean(lda.y!=lda.predy)
      #error rates
      CV=c(CV,error)
    }
    #Output
    list(call = model, K = K,
         lda_error_rate = mean(CV), seed = seed)
  }
er_lda=cv.lda(Data,model=Legendary~Attack+Defense+Sp..Atk+Sp..De
f+Speed, yname="Legendary", K=10, seed=1)
cat("The test error for LDA 10-fold cross validation
is:",er_lda$lda_error_rate)


######################
# Validation Set LDA #
######################
```

```
set.seed(1)

n <- nrow(Data)

train=sample(1:n,n/2)

train.set=Data[train,]

test.set=Data[-train,]

library(MASS)

lda.fit=lda(Legendary~Attack+Defense+Sp..Atk+Sp..Def+Speed,data=
train.set)

predict.fit=predict(lda.fit,test.set)

MSE=mean((predict.fit$class!=test.set$Legendary))

cat("The test error for LDA validation set is:",MSE)


######################

# Validation Set KNN #

######################

set.seed(1)

n <- nrow(Data)

train=sample(1:n,n/2)

train.set=Data[train,]

test.set=Data[-train,]

library(MASS)

features=c("Attack","Defense","Sp..Atk","Sp..Def","Speed")

knn.fit=knn(train.set[,features],test.set[,features],train.set$L
egendary,k=6)

MSE=mean(knn.fit!=test.set$Legendary)

cat("The test error for KNN validation set is:",MSE)

#Prediction for Type.2

---
```

```
title: "pre_type2"
author: "Xuwen Shen  xs288"
date: "11/30/2017"
output:
  word_document: default
  html_document: default
---


Prediction to type.2

```{r, echo=FALSE}
data_p = read.csv("Pokemon.csv", header = T, sep = ",", na.strings = "NA")


#fit1 = lm(Type.1~., data = data_p)


str(data_p)
data_p$Type.1 = as.numeric(data_p$Type.1)
#data_p$Legendary = as.numeric(data_p$Legendary)
fit1 = lm(Type.1~Total+HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Generation+factor(Legendary), data = data_p)
length(data_p[,1])
```

```{r,echo=FALSE}
error_l = rep(0,10)
error_d = rep(0,10)
error_rf = rep(0,10)
error_k = rep(0,10)
```
```

```{r}
#transfer type.2 na to na
data_p$Type.2[data_p$Type.2==""]="none"
str(data_p$Type.2)
```

Question 1.

1) Prediction for Type.2

We plotted boxplot for all the predictors to show the variration for the predictors.

From the plot we can see, the predictor: HP,Total,Attack, Defense, Sp..Atk, Speed has significant variation, so we initially choose HP,Total,Attack, Defense, Sp..Atk, Speed.

And we delete the predictor Total due to the coliearity.

```{r,echo=FALSE}
boxplot(HP~Type.2, data = data_p,  log = "y", col = "bisque")
boxplot(Total~Type.2, data = data_p)#
boxplot(Attack~Type.2, data = data_p,main ="Attack")#
boxplot(Defense~Type.2, data = data_p,main="Defense")#
boxplot(Sp..Atk~Type.2, data = data_p,main="Sp..Atk")#
boxplot(Sp..Def~Type.2, data = data_p)
boxplot(Speed~Type.2,data = data_p,main="Speed")#
boxplot(Generation~Type.2,data = data_p)
#boxplot(Legendary~factor(Type.1),data = data
```

Method 1. LDA

```{r,echo=FALSE}

library(MASS)

# Validation set

#set.seed(1)

na_ind = is.na(data_p$Type.2)

#data_p$Type.2[na_ind] = "N"

set.seed(1)

train400=sample(1:800,400)

train=sample(1:800,400)

train_data = data_p[train400,]

test_data = data_p[-train400,]

train_y = data_p$Type.2[train400]

test_y = data_p$Type.2[-train400]

fit_lda = lda(Type.2~HP+Attack+Defense+Sp..Atk+Speed, data = data_p, subset = train)

#fit_lda

pre_lda = predict(fit_lda, test_data)

lda.class = pre_lda$class

error_l_type2 = mean(lda.class != test_y,na.rm = T);error_l_type2

```

With only the variable we selected with boxplot, we get the value of 0.71 for test error.

We used the same method in prediction for Type.2.

Subset Selection

```{r}

##########################################################
```

```
# ***************** NEW SECTION ADDED ***************** #
# *************** Selecting Parameters *************** #
#                                                       #
# ************** ADD ME TO THE FULL CODE ************** #
#########################################################
error_l_SS_a=matrix(0,nrow=10,ncol=8)


#All Variables (Not Including Total)
#full.fit=glm(Type.2~HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Gen
eration+Legendary, data = data_p)
sample_vec = c(1:10)
fold_indeces <- vector("list", 10)
  for(i in 1:10){
    fold_index_test <- sample(1:800, 80)
    fold_indeces[[i]] <- fold_index_test
    sample_vec <- sample_vec[-fold_index_test]
  }


error_l_SS_a = c(rep(0,10))
for (j in 1:10) {
    test <- fold_indeces[[j]]
    train = c(1:800)[-test]
    dat_test <- data_p[test,]
    dat_train <- data_p[-test,]
    test_y = dat_test$Type.2
    # LDA
    full_lda                                                    =
lda(data_p$Type.2~HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Genera
tion+Legendary, data = data_p, subset = train)
```

```
    pre_full_lda = predict(full_lda, dat_test)


    error_l_SS_a[j] = mean(pre_full_lda$class != test_y,na.rm =
T)
}


mean(error_l_SS_a)
```

So the full model as error rate of 0.784935.

Remove 1 variable

```{r,echo=FALSE}
#############################################################
# ***************** NEW SECTION ADDED ***************** #
# *************** Selecting Parameters *************** #
#                                                       #
# ************** ADD ME TO THE FULL CODE ************** #
#############################################################



#All Variables (Not Including Total)
#full.fit=glm(Type.2~HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Gen
eration+Legendary, data = data_p)
matrix.Errors=matrix(0,nrow=50,ncol=8)
for (i in 1:50){
  set.seed(i)
  sample_vec = c(1:10)
  fold_indeces <- vector("list", 10)
```

```
    for(h in 1:10){
      fold_index_test <- sample(1:800, 80)
      fold_indeces[[h]] <- fold_index_test
      sample_vec <- sample_vec[-fold_index_test]
    }

  error_l_SS_8=matrix(0,nrow=10,ncol=8)
  for (j in 1:10) {
      test <- fold_indeces[[j]]
      train = c(1:800)[-test]
      dat_test <- data_p[test,]
      dat_train <- data_p[-test,]
      test_y = dat_test$Type.1

      # LDA

      #Remove HP
      lda.HP                                                    =
lda(data_p$Type.1~Attack+Defense+Sp..Atk+Sp..Def+Speed+Generatio
n+Legendary, data = data_p, subset = train)
      pre_lda.HP = predict(lda.HP, dat_test)
      error_l_SS_8[j,1] = mean(pre_lda.HP$class != test_y)

      #Remove Attack
      lda.Attack                                                =
lda(data_p$Type.1~HP+Defense+Sp..Atk+Sp..Def+Speed+Generation+Le
gendary, data = data_p, subset = train)
      pre_lda.Attack = predict(lda.Attack, dat_test)
      error_l_SS_8[j,2] = mean(pre_lda.Attack$class != test_y)
```

```
      #Remove Defense

      lda.Defense                                                    =
lda(data_p$Type.1~HP+Attack+Sp..Atk+Speed+Sp..Def+Generation+Leg
endary, data = data_p, subset = train)

      pre_lda.Defense = predict(lda.Defense, dat_test)

      error_1_SS_8[j,3] = mean(pre_lda.Defense$class != test_y)


      #Remove Sp..Atk

      lda.Sp..Atk                                                    =
lda(data_p$Type.1~HP+Attack+Defense+Speed+Sp..Def+Generation+Leg
endary, data = data_p, subset = train)

      pre_lda.Sp..Atk = predict(lda.Sp..Atk, dat_test)

      error_1_SS_8[j,4] = mean(pre_lda.Sp..Atk$class != test_y)


      #Remove Sp..De

      lda.Speed                                                      =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Speed+Generation+Leg
endary, data = data_p, subset = train)

      pre_lda.Speed = predict(lda.Speed, dat_test)

      error_1_SS_8[j,5] = mean(pre_lda.Speed$class != test_y)


      #Remove Speed

      lda.Speed                                                      =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Sp..Def+Generation+L
egendary, data = data_p, subset = train)

      pre_lda.Speed = predict(lda.Speed, dat_test)

      error_1_SS_8[j,6] = mean(pre_lda.Speed$class != test_y)


      #Remove Generation
```

```
      lda.Gen                                                    =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Legend
ary, data = data_p, subset = train)

      pre_lda.Gen = predict(lda.Gen, dat_test)

      error_1_SS_8[j,7] = mean(pre_lda.Gen$class != test_y)


      #Remove Legendary

      lda.Leg                                                    =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Genera
tion, data = data_p, subset = train)

      pre_lda.Leg = predict(lda.Leg, dat_test)

      error_1_SS_8[j,8] = mean(pre_lda.Leg$class != test_y)

  }

  for (k in 1:8){

    matrix.Errors[i,k]=mean(error_1_SS_8[,k])

  }

}

result=matrix(c("HP","Attack","Defense","Sp..Atk","Sp..De","Spee
d","Generation","Legendary",mean(matrix.Errors[,1]),mean(matrix.
Errors[,2]),mean(matrix.Errors[,3]),mean(matrix.Errors[,4]),mean
(matrix.Errors[,5]),mean(matrix.Errors[,6]),mean(matrix.Errors[,
7]),mean(matrix.Errors[,8])),ncol=2,nrow=8)

num=which(grepl(min(result[,2]), result[,2]))

result[num,]
```

In this step, we remove one single predictor from the 8 predictors and we found that we get the smallest cross validation test error with the value of 0.774875, if we romoved Sp..De. So we remove Sp.De for this step.

```{r,echo=FALSE}
```

```
result=matrix(c("HP","Attack","Defense","Sp..Atk","Sp..Def","Spe
ed","Generation","Legendary",mean(matrix.Errors[,1]),mean(matrix
.Errors[,2]),mean(matrix.Errors[,3]),mean(matrix.Errors[,4]),mea
n(matrix.Errors[,5]),mean(matrix.Errors[,6]),mean(matrix.Errors[
,7]),mean(matrix.Errors[,8])),ncol=2,nrow=8)

num=which(grepl(min(result[,2]), result[,2]))

result[num,]

df_l_8 = data.frame("Re.HP" = matrix.Errors[,1],"Re.Atta" =
matrix.Errors[,2],"Re.Def" = matrix.Errors[,3],"ReSp..Atk" =
matrix.Errors[,4],"RE.Sp..De"=matrix.Errors[,5],"Re.Speed" =
matrix.Errors[,6],"Re.Ge"=matrix.Errors[,7],"Re.Le"=matrix.Error
s[,8])

boxplot(df_l_8,main= "Boxplot with seeds 1-
50",xlab="Variables",ylab="Test error")
```

Thus, remove special defense.

****
****

****
****

Now models with 2 predictors removed.

```{r,echo=FALSE}

##########
```

```
# ***************** NEW SECTION ADDED ***************** #
# *************** Selecting Parameters *************** #
#                                                       #
# ************** ADD ME TO THE FULL CODE ************** #
#########################################################


matrix.Errors=matrix(0,nrow=50,ncol=7)
for (i in 1:50){
  set.seed(i)
  sample_vec = c(1:10)
  fold_indeces <- vector("list", 10)
    for(h in 1:10){
      fold_index_test <- sample(1:800, 80)
      fold_indeces[[h]] <- fold_index_test
      sample_vec <- sample_vec[-fold_index_test]
  }

  error_l_SS_7=matrix(0,nrow=10,ncol=7)
  for (j in 1:10) {
      test <- fold_indeces[[j]]
      train = c(1:800)[-test]
      dat_test <- data_p[test,]
      dat_train <- data_p[-test,]
      test_y = dat_test$Type.1

      # LDA
```

```
      #Remove HP
      lda.HP                                                    =
lda(data_p$Type.1~Attack+Defense+Sp..Atk+Speed+Generation+Legend
ary, data = data_p, subset = train)
      pre_lda.HP = predict(lda.HP, dat_test)
      error_1_SS_7[j,1] = mean(pre_lda.HP$class != test_y)


      #Remove Attack
      lda.Attack                                                =
lda(data_p$Type.1~HP+Defense+Sp..Atk+Speed+Generation+Legendary,
data = data_p, subset = train)
      pre_lda.Attack = predict(lda.Attack, dat_test)
      error_1_SS_7[j,2] = mean(pre_lda.Attack$class != test_y)


      #Remove Defense
      lda.Defense                                               =
lda(data_p$Type.1~HP+Attack+Sp..Atk+Speed+Generation+Legendary,
data = data_p, subset = train)
      pre_lda.Defense = predict(lda.Defense, dat_test)
      error_1_SS_7[j,3] = mean(pre_lda.Defense$class != test_y)


      #Remove Sp..Atk
      lda.Sp..Atk                                               =
lda(data_p$Type.1~HP+Attack+Defense+Speed+Generation+Legendary,
data = data_p, subset = train)
      pre_lda.Sp..Atk = predict(lda.Sp..Atk, dat_test)
      error_1_SS_7[j,4] = mean(pre_lda.Sp..Atk$class != test_y)


      #Remove Speed
```

```
      lda.Speed                                                    =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Generation+Legendary
, data = data_p, subset = train)

      pre_lda.Speed = predict(lda.Speed, dat_test)

      error_l_SS_7[j,5] = mean(pre_lda.Speed$class != test_y)


      #Remove Generation

      lda.Gen                                                      =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Speed+Legendary,
data = data_p, subset = train)

      pre_lda.Gen = predict(lda.Gen, dat_test)

      error_l_SS_7[j,6] = mean(pre_lda.Gen$class != test_y)


      #Remove Legendary

      lda.Leg                                                      =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Speed+Generation,
data = data_p, subset = train)

      pre_lda.Leg = predict(lda.Leg, dat_test)

      error_l_SS_7[j,7] = mean(pre_lda.Leg$class != test_y)

  }

  for (k in 1:7){

    matrix.Errors[i,k]=mean(error_l_SS_7[,k])

  }

}

result=matrix(c("HP","Attack","Defense","Sp..Atk","Speed","Gener
ation","Legendary",mean(matrix.Errors[,1]),mean(matrix.Errors[,2
]),mean(matrix.Errors[,3]),mean(matrix.Errors[,4]),mean(matrix.E
rrors[,5]),mean(matrix.Errors[,6]),mean(matrix.Errors[,7])),ncol
=2,nrow=7)

num=which(grepl(min(result[,2]), result[,2]))

result[num,]
```

In this step, we remove one single predictor from the 7 predictors and we found that we get the smallest cross validation test error with the value of 0.76985, if we romoved Generation So we remove Generation for this step.

```{r}

df_l_7 = data.frame("Re.HP" = matrix.Errors[,1],"Re.Atta" = matrix.Errors[,2],"Re.Def" = matrix.Errors[,3],"ReSp..Atk" = matrix.Errors[,4],"Re.Speed" = matrix.Errors[,5],"Re.Ge"=matrix.Errors[,6],"Re.Le"=matrix.Errors[,7])

boxplot(df_l_7,main= "Boxplot with seeds 1-50",xlab="Variables",ylab="Test error")

#boxplot(matrix.Errors[,1],matrix.Errors[,2],matrix.Errors[,3],matrix.Errors[,4],matrix.Errors[,5],matrix.Errors[,#6],matrix.Errors[,7],xlab="Predictors moved",ylab="Test error")

```

So gerneration is removed.

Now model with 3 variables removed.

```{r,echo=FALSE}

############################################################

# ***************** NEW SECTION ADDED ***************** #

# *************** Selecting Parameters *************** #

# #

# ************** ADD ME TO THE FULL CODE ************** #

############################################################



matrix.Errors=matrix(0,nrow=50,ncol=6)

for (i in 1:50){

  set.seed(i)

  sample_vec = c(1:10)

  fold_indeces <- vector("list", 10)
```

```
    for(h in 1:10){
      fold_index_test <- sample(1:800, 80)
      fold_indeces[[h]] <- fold_index_test
      sample_vec <- sample_vec[-fold_index_test]
    }

  error_l_SS=matrix(0,nrow=10,ncol=6)
  for (j in 1:10) {
      test <- fold_indeces[[j]]
      train = c(1:800)[-test]
      dat_test <- data_p[test,]
      dat_train <- data_p[-test,]
      test_y = dat_test$Type.1

      # LDA

      #Remove HP
      lda.HP                                                    =
lda(data_p$Type.1~Attack+Defense+Sp..Atk+Speed+Legendary,  data  =
data_p, subset = train)
      pre_lda.HP = predict(lda.HP, dat_test)
      error_l_SS[j,1] = mean(pre_lda.HP$class != test_y)

      #Remove Attack
      lda.Attack                                                =
lda(data_p$Type.1~HP+Defense+Sp..Atk+Speed+Legendary,    data    =
data_p, subset = train)
      pre_lda.Attack = predict(lda.Attack, dat_test)
      error_l_SS[j,2] = mean(pre_lda.Attack$class != test_y)
```

```
      #Remove Defense
      lda.Defense                                                    =
lda(data_p$Type.1~HP+Attack+Sp..Atk+Speed+Legendary,    data    =
data_p, subset = train)
      pre_lda.Defense = predict(lda.Defense, dat_test)
      error_l_SS[j,3] = mean(pre_lda.Defense$class != test_y)


      #Remove Sp..Atk
      lda.Sp..Atk                                                    =
lda(data_p$Type.1~HP+Attack+Defense+Speed+Legendary,    data    =
data_p, subset = train)
      pre_lda.Sp..Atk = predict(lda.Sp..Atk, dat_test)
      error_l_SS[j,4] = mean(pre_lda.Sp..Atk$class != test_y)


      #Remove Speed
      lda.Speed                                                      =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Legendary,   data   =
data_p, subset = train)
      pre_lda.Speed = predict(lda.Speed, dat_test)
      error_l_SS[j,5] = mean(pre_lda.Speed$class != test_y)


      #Remove Legendary
      lda.Leg                                                        =
lda(data_p$Type.1~HP+Attack+Defense+Sp..Atk+Speed,     data     =
data_p, subset = train)
      pre_lda.Leg = predict(lda.Leg, dat_test)
      error_l_SS[j,6] = mean(pre_lda.Leg$class != test_y)
  }
  for (k in 1:6){
    matrix.Errors[i,k]=mean(error_l_SS[,k])
```

```
  }
}
result=matrix(c("HP","Attack","Defense","Sp..Atk","Speed","Legendary",mean(matrix.Errors[,1]),mean(matrix.Errors[,2]),mean(matrix.Errors[,3]),mean(matrix.Errors[,4]),mean(matrix.Errors[,5]),mean(matrix.Errors[,6])),ncol=2,nrow=6)
num=which(grepl(min(result[,2]), result[,2]))
result[num,]
```

In this step, we remove one single predictor from the 7 predictors and we found that we get the smallest cross validation test error with the value of 0.7744, if we romoved Legendary So we remove Legendary for this step.

Thus, remove HP.

```{r}
############################################################
# ***************** NEW SECTION ADDED ***************** #
# *************** Selecting Parameters *************** #
#                                                          #
# ************** ADD ME TO THE FULL CODE ************** #
############################################################



matrix.Errors=matrix(0,nrow=50,ncol=6)
for (i in 1:50){
  set.seed(i)
  sample_vec = c(1:10)
  fold_indeces <- vector("list", 10)
    for(h in 1:10){
      fold_index_test <- sample(1:800, 80)
```

```
      fold_indeces[[h]] <- fold_index_test
      sample_vec <- sample_vec[-fold_index_test]
  }

  error_l_SS=matrix(0,nrow=10,ncol=6)
  for (j in 1:10) {
      test <- fold_indeces[[j]]
      train = c(1:800)[-test]
      dat_test <- data_p[test,]
      dat_train <- data_p[-test,]
      test_y = dat_test$Type.2

      # LDA

      #Remove HP
      lda.HP                                                    =
lda(data_p$Type.2~Attack+Defense+Sp..Atk+Speed+Legendary,  data  =
data_p, subset = train)
      pre_lda.HP = predict(lda.HP, dat_test)
      error_l_SS[j,1] = mean(pre_lda.HP$class != test_y)

      #Remove Attack
      lda.Attack                                                =
lda(data_p$Type.2~HP+Defense+Sp..Atk+Speed+Legendary,    data    =
data_p, subset = train)
      pre_lda.Attack = predict(lda.Attack, dat_test)
      error_l_SS[j,2] = mean(pre_lda.Attack$class != test_y)

      #Remove Defense
```

```
      lda.Defense                                                    =
lda(data_p$Type.2~HP+Attack+Sp..Atk+Speed+Legendary,    data    =
data_p, subset = train)

      pre_lda.Defense = predict(lda.Defense, dat_test)

      error_l_SS[j,3] = mean(pre_lda.Defense$class != test_y)


      #Remove Sp..Atk

      lda.Sp..Atk                                                    =
lda(data_p$Type.2~HP+Attack+Defense+Speed+Legendary,    data    =
data_p, subset = train)

      pre_lda.Sp..Atk = predict(lda.Sp..Atk, dat_test)

      error_l_SS[j,4] = mean(pre_lda.Sp..Atk$class != test_y)


      #Remove Speed

      lda.Speed                                                      =
lda(data_p$Type.2~HP+Attack+Defense+Sp..Atk+Legendary,   data   =
data_p, subset = train)

      pre_lda.Speed = predict(lda.Speed, dat_test)

      error_l_SS[j,5] = mean(pre_lda.Speed$class != test_y)


      #Remove Legendary

      lda.Leg                                                        =
lda(data_p$Type.2~HP+Attack+Defense+Sp..Atk+Speed,     data     =
data_p, subset = train)

      pre_lda.Leg = predict(lda.Leg, dat_test)

      error_l_SS[j,6] = mean(pre_lda.Leg$class != test_y)

  }

  for (k in 1:6){

    matrix.Errors[i,k]=mean(error_l_SS[,k])

  }

}
```

```
result=matrix(c("HP","Attack","Defense","Sp..Atk","Speed","Legen
dary",mean(matrix.Errors[,1]),mean(matrix.Errors[,2]),mean(matri
x.Errors[,3]),mean(matrix.Errors[,4]),mean(matrix.Errors[,5]),me
an(matrix.Errors[,6])),ncol=2,nrow=6)

num=which(grepl(min(result[,2]), result[,2]))

result[num,]
```

```{r}
############################################################

# ***************** NEW SECTION ADDED ***************** #

# *************** Selecting Parameters *************** #

#                                                        #

# ************** ADD ME TO THE FULL CODE ************** #

############################################################



matrix.Errors=matrix(0,nrow=50,ncol=5)

for (i in 1:50){

  set.seed(i)

  sample_vec = c(1:10)

  fold_indeces <- vector("list", 10)

    for(h in 1:10){

      fold_index_test <- sample(1:800, 80)

      fold_indeces[[h]] <- fold_index_test

      sample_vec <- sample_vec[-fold_index_test]

  }


  error_l_SS=matrix(0,nrow=10,ncol=5)

  for (j in 1:10) {
```

```
      test <- fold_indeces[[j]]
      train = c(1:800)[-test]
      dat_test <- data_p[test,]
      dat_train <- data_p[-test,]
      test_y = dat_test$Type.2

      # LDA

      #Remove HP
      lda.HP   =   lda(data_p$Type.2~Attack+Defense+Sp..Atk+Speed,
data = data_p, subset = train)
      pre_lda.HP = predict(lda.HP, dat_test)
      error_1_SS[j,1] = mean(pre_lda.HP$class != test_y)

      #Remove Attack
      lda.Attack   =   lda(data_p$Type.2~HP+Defense+Sp..Atk+Speed,
data = data_p, subset = train)
      pre_lda.Attack = predict(lda.Attack, dat_test)
      error_1_SS[j,2] = mean(pre_lda.Attack$class != test_y)

      #Remove Defense
      lda.Defense   =   lda(data_p$Type.2~HP+Attack+Sp..Atk+Speed,
data = data_p, subset = train)
      pre_lda.Defense = predict(lda.Defense, dat_test)
      error_1_SS[j,3] = mean(pre_lda.Defense$class != test_y)

      #Remove Sp..Atk
      lda.Sp..Atk   =   lda(data_p$Type.2~HP+Attack+Defense+Speed,
data = data_p, subset = train)
```

```
      pre_lda.Sp..Atk = predict(lda.Sp..Atk, dat_test)
      error_l_SS[j,4] = mean(pre_lda.Sp..Atk$class != test_y)


      #Remove Speed
      lda.Speed   =  lda(data_p$Type.2~HP+Attack+Defense+Sp..Atk,
data = data_p, subset = train)
      pre_lda.Speed = predict(lda.Speed, dat_test)
      error_l_SS[j,5] = mean(pre_lda.Speed$class != test_y)
  }
  for (k in 1:5){
    matrix.Errors[i,k]=mean(error_l_SS[,k])
  }
}
result=matrix(c("HP","Attack","Defense","Sp..Atk","Speed",mean(m
atrix.Errors[,1]),mean(matrix.Errors[,2]),mean(matrix.Errors[,3]
),mean(matrix.Errors[,4]),mean(matrix.Errors[,5])),ncol=2,nrow=5
)
num=which(grepl(min(result[,2]), result[,2]))
result[num,]
```

```{r}
#############################################################
# ***************** NEW SECTION ADDED ***************** #
# *************** Selecting Parameters *************** #
#                                                       #
# ************** ADD ME TO THE FULL CODE ************** #
#############################################################
```

```
matrix.Errors=matrix(0,nrow=50,ncol=4)
for (i in 1:50){
  set.seed(i)
  sample_vec = c(1:10)
  fold_indeces <- vector("list", 10)
    for(h in 1:10){
      fold_index_test <- sample(1:800, 80)
      fold_indeces[[h]] <- fold_index_test
      sample_vec <- sample_vec[-fold_index_test]
  }

  error_l_SS=matrix(0,nrow=10,ncol=4)
  for (j in 1:10) {
      test <- fold_indeces[[j]]
      train = c(1:800)[-test]
      dat_test <- data_p[test,]
      dat_train <- data_p[-test,]
      test_y = dat_test$Type.2

      # LDA

      #Remove Attack
      lda.Attack = lda(data_p$Type.2~Defense+Sp..Atk+Speed, data
= data_p, subset = train)
      pre_lda.Attack = predict(lda.Attack, dat_test)
      error_l_SS[j,1] = mean(pre_lda.Attack$class != test_y)
```

```
      #Remove Defense

      lda.Defense = lda(data_p$Type.2~Attack+Sp..Atk+Speed, data
= data_p, subset = train)

      pre_lda.Defense = predict(lda.Defense, dat_test)

      error_l_SS[j,2] = mean(pre_lda.Defense$class != test_y)


      #Remove Sp..Atk

      lda.Sp..Atk = lda(data_p$Type.2~Attack+Defense+Speed, data
= data_p, subset = train)

      pre_lda.Sp..Atk = predict(lda.Sp..Atk, dat_test)

      error_l_SS[j,3] = mean(pre_lda.Sp..Atk$class != test_y)


      #Remove Speed

      lda.Speed = lda(data_p$Type.2~Attack+Defense+Sp..Atk, data
= data_p, subset = train)

      pre_lda.Speed = predict(lda.Speed, dat_test)

      error_l_SS[j,4] = mean(pre_lda.Speed$class != test_y)

  }

  for (k in 1:4){

    matrix.Errors[i,k]=mean(error_l_SS[,k])

  }

}

result=matrix(c("Attack","Defense","Sp..Atk","Speed",mean(matrix
.Errors[,1]),mean(matrix.Errors[,2]),mean(matrix.Errors[,3]),mea
n(matrix.Errors[,4])),ncol=2,nrow=4)

num=which(grepl(min(result[,2]), result[,2]))

result[num,]
```

Thus, remove Speed.

Now remove 6 parameters and model.

```{r}
#########################################################
# ***************** NEW SECTION ADDED ***************** #
# *************** Selecting Parameters *************** #
#                                                       #
# ************** ADD ME TO THE FULL CODE ************** #
#########################################################


matrix.Errors=matrix(0,nrow=50,ncol=3)
for (i in 1:50){
  set.seed(i)
  sample_vec = c(1:10)
  fold_indeces <- vector("list", 10)
    for(h in 1:10){
      fold_index_test <- sample(1:800, 80)
      fold_indeces[[h]] <- fold_index_test
      sample_vec <- sample_vec[-fold_index_test]
  }

```

Tree

```{r,echo=FALSE}
library(tree)
```

```
library(ISLR)

Type.1_s = factor(data_p$Type.2)

tree.fit=tree(Type.2~HP+Attack+Defense+Sp..Atk+Speed+Legendary,d
ata_p,subset = train)

#summary(tree.fit)

tree.pred=predict(tree.fit,test_data,type="class")

mean(tree.pred != test_y,na.rm = T)
```