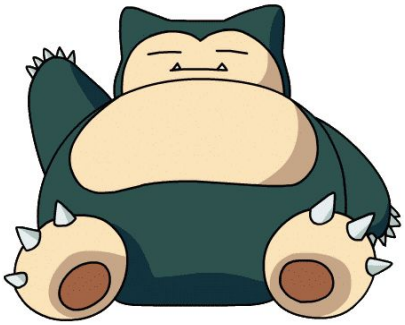
4740 Final Project Report

Fall 2017
Instructor: Yang Ning

Students:

Xu Shen xs286
Yuxi Chen yc2435
Xiaoyan Zhu xz568
Shuai Qu sq74



12/03/2017

1. Exploratory Data Analysis

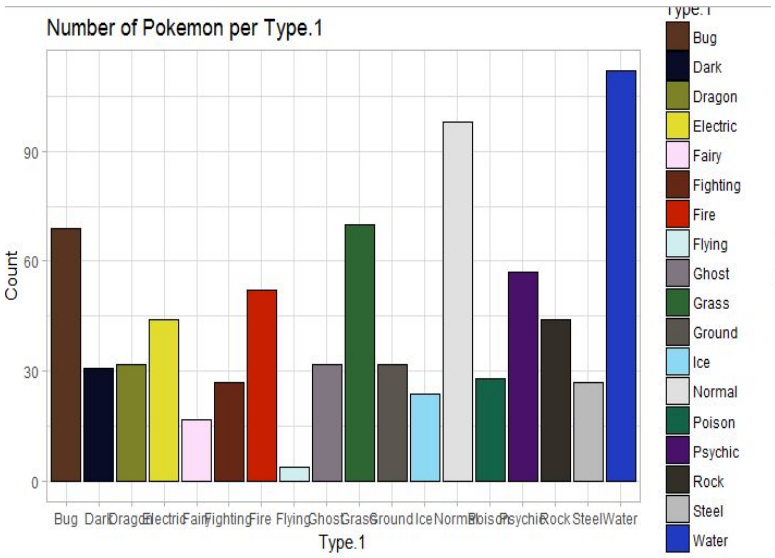
Missing Value Checking

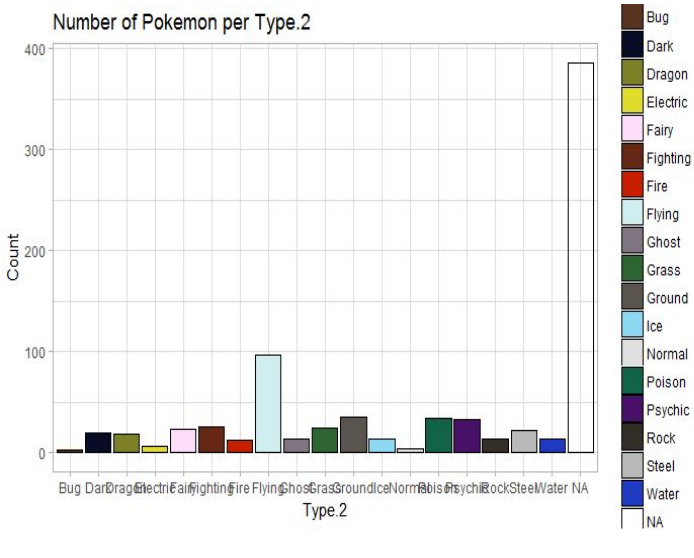
The Pokémon dataset table consists of 800 rows and 13 columns. We initially checked the existence of missing values and later found most null values appear in Type 2.

Distribution of Pokémon Types

By exploring the distributions of Type 1 and Type 2 of the Pokémon as shown below, we discovered that

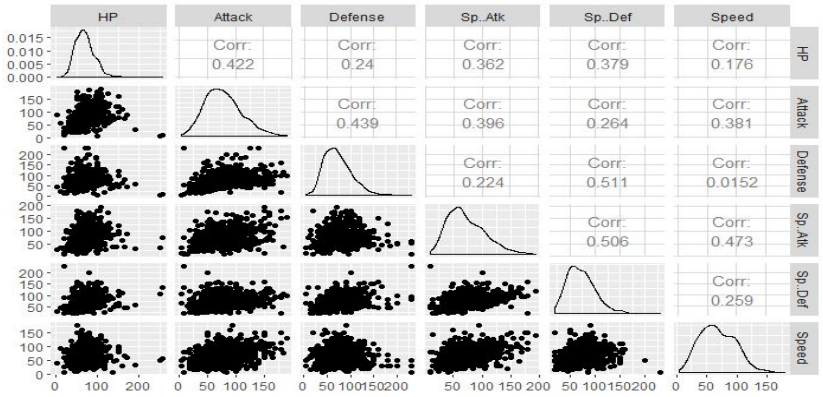
- The most frequent Type 1 category is Water while Flying is the least frequent Type 1 category.
- Nearly half (386) of the Pokémon has no dual type (no Type 2).
- Nearly 1 out of 7 Pokémon have a secondary flying-type.





Collinearity Check

We then conducted a pairwise comparisons of the statistics of the Pokémon and applied the collinearity check.



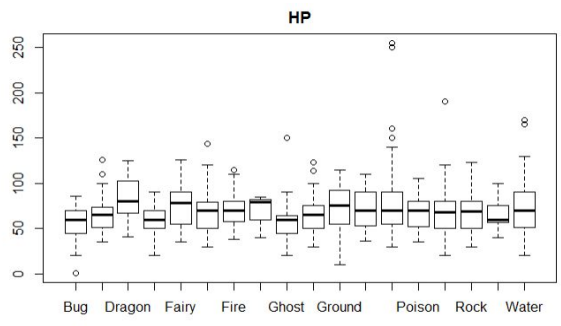
The covariance matrix plot shown above indicates that almost all the statistics are correlated except for the defense and speed. Hence, we decided not to use linear models to tackle this problem.

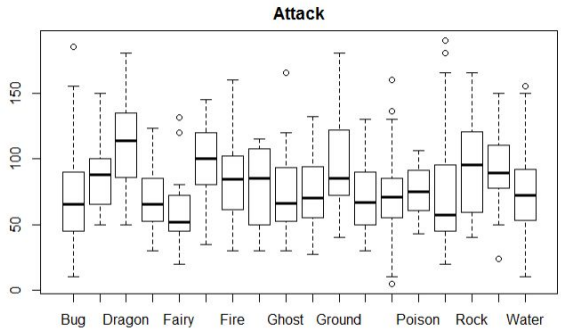
The most correlated three statistics are (in order):

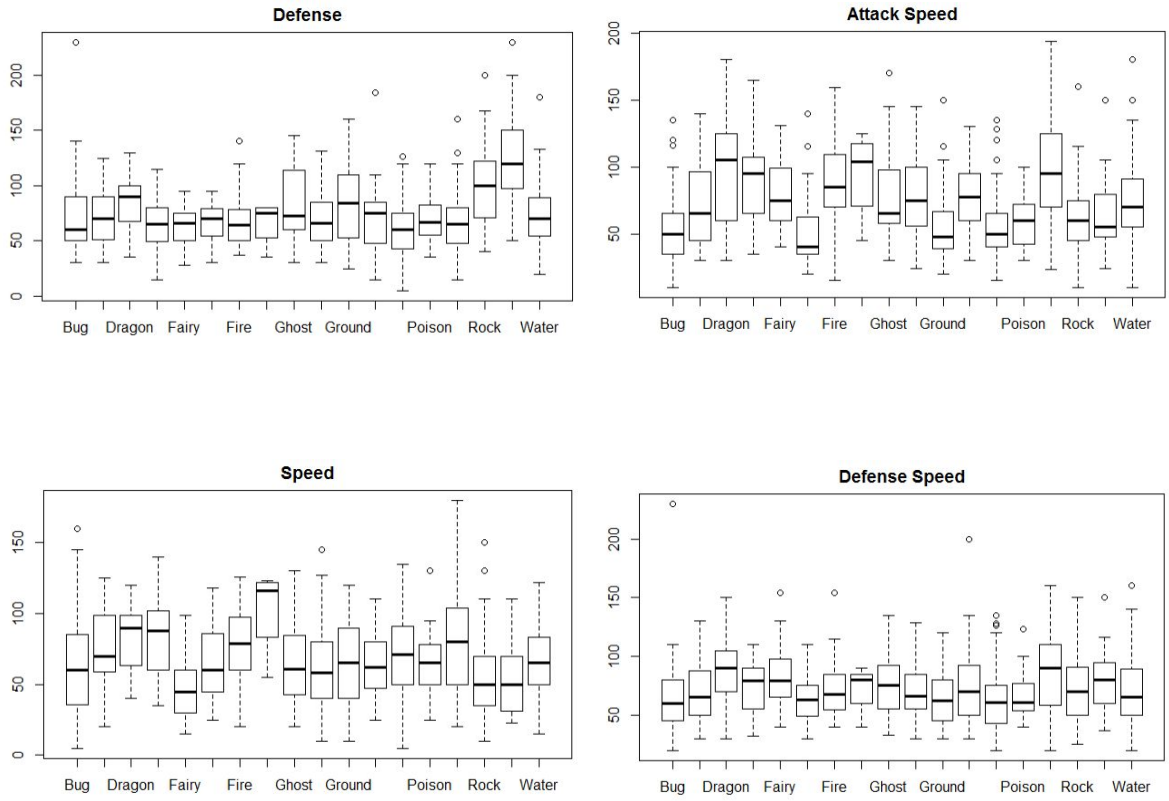
- speed defense/defense
- speed attack/speed defense
- speed/speed attack

Hypothesis

By plotting different statistics of Pokémon vs. Type 1 as below, we discovered that the attack speed, attack, speed and defense vary a lot among different types of Pokémon. Thus, we made a hypothesis that the total, attack speed, attack, speed and defense can define the type of Pokémon.







2. Type Classification

In this problem, we used 10-fold cross validation mean error rate as metrics to perform model selection and parameter tuning. We would build statistical learning models on classifying the primary type (Type 1) of the Pokémon and evaluate variables that can mainly define the type of a Pokémon. Then, we continue classifying the combination of two type.

2.1 Classification Model on Type1 Pokémon

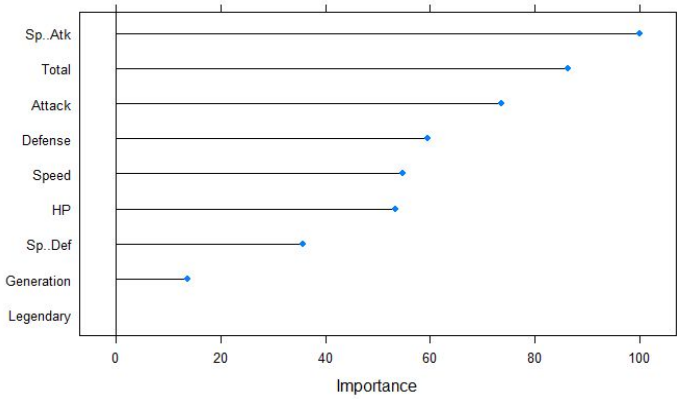
2.1.1 Boosting Tree model

We initially designed the boosted classification tree models to perform classification on the primary type (Type1) of Pokémon with all the variables as predictors excluding variable Name, Type1 and Type2.

Later, the Mean error rate of our model is 75.27%. 10-fold cross validation error rate is computed to select the tuning parameters by the lowest value. The number of trees is 50 and the number of splits in each tree is 3, while the shrinkage parameter is held constant at 0.1.

The varImp function is exercised to rank variables predicting Type 1 in this model from more to less importance as followings:

Sp..Atk>Total>Attack>Defense>Speed>HP>Sp..Def>Generation>Legendary



2.1.2 Random Forest Model

A random forest model was also designed to predict the primary type of pokemon. We assigned 10-fold cross validation to determine the parameter mtry and had mtry=2 with the number of trees equals to 500. The mean error rate is 75.93%.

Again, the varImp function was implemented to rank variables predicting Type 1 in this model from more to less importance as followings:

Sp..Atk>Total>Attack>Speed>Defense>HP>Sp..Def>Generation>Legendary

2.1.3 Model Selection

In this problem, the classification boundaries are very likely to be nonlinear. Hence, we chose tree-based models based on their hierarchical structure. According to the mean error rate, the boosting tree model outperformed the random forest model by 0.7%. The difference is very small. We tended to conduct boosted tree methods over random forest for the following reasons:

Bias-variance trade off issue: Random forest is less likely to overfit than boosted tree models. However, in this case, the variance is not our priority. Since the model will not classify on other sets in the real word. To achieve low bias, we choose boosted tree methods, because, Boosted Trees(GBM) tries to add new trees that compliments the already built ones. This normally gives our better accuracy with less trees comparing to the random forest.

The importance of variables is ranked from high to low as follows:

Sp..Atk>Total>Attack>Defense>Speed>HP>Sp..Def>Generation>Legendary

Therefore, we selected the top five attributes from the above ranking as our answer to the question 1. They are Attack speed, Total sum of stats, Attack stats, Defense stats and Speed. This is almost in compliance with our preliminary hypothesis.

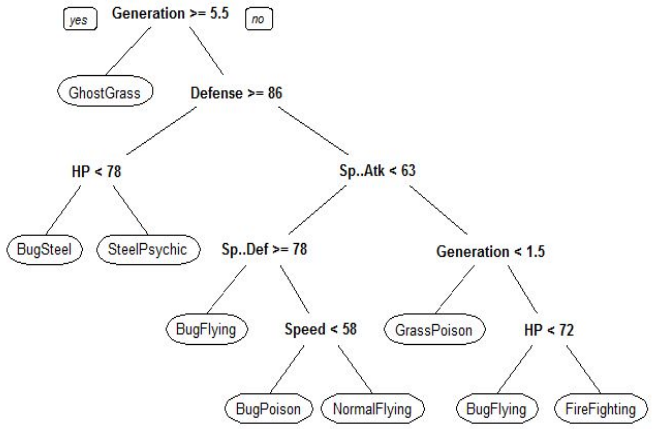
2.2 Combination of Two Types

In this part, we would further investigate the combination of two types. We grouped all the Pokémon according to Type 1 and Type 2 and rank them by descending. The top 10 combinations of types of Pokémon are shown as below:

Type1	Type2	Numbers
Normal	Flying	24
Grass	Poison	15
Bug	Flying	14
Bug	Poison	12
Ghost	Grass	10
Water	Ground	10
Bug	Steel	7
Fire	Fighting	7
Steel	Psychic	7
Water	Flying	7

In this project, we inserted a new feature 'DualType' in the dataset and was only interested in the top 10 combinations of Type1 and Type 2 of Pokémon, while the rest of the 'DualType' attributes would be treated as null values.

Moreover, we built a tree model as below to explore the classification structure of the combination of the two types:



We learned that

- Ghost-Grass dual type Pokémon has the latest generation.
- Bug-Steel and Steel-Psychic dual type tend to get low defense value.
- In the terms of Attack-Speed, the Normal-Flying dual type Pokémon is slower than the Fire-Fighting Pokémon.

We conclude that the generation, defense and attack speed can define the combination types of the Pokémon from the tree diagram.

3. Legendary Classification

3.1 Logistic Regression Model

Since the dependent variable Legendary is a binary variable, a logistic regression can be built to predict the Legendary. We preferred Type.1, Type.2, Total, HP, Attack, Defense, Sp.AtK,

Sp.Def, Speed and Generation as the predictors in our logistic regression model. By calculating a 10-fold cross validation error, we got the mean error rate of our model is 0.06625.

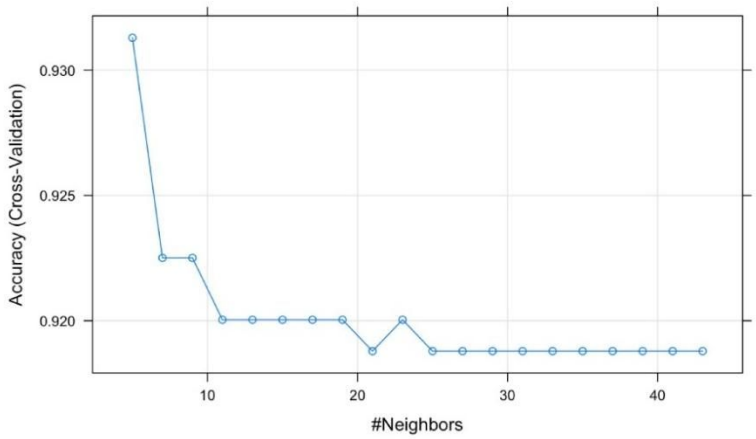
Although the logistic regression owns a good probabilistic interpretation, it tends to underperform when there are multiple or nonlinear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

In this Pokémon problem, we suspected that the decision boundaries are highly non-linear decision boundaries. Therefore, we would further try the KNN and QDA, which are expected to perform better on non-linear decision boundaries problems.

3.2 K-Nearest Neighbors

For the Pokémon dataset, we fit the KNN model with all qualitative and quantitative variables since it does not need any assumptions about the shape of the decision boundary.

We produced a Cross-Validation Error curve to illustrate the performance of KNN method. According to the cv error curve below, we recognized that the highest accuracy is around 0.93, and the corresponding k value is about 4 or 5. After examining the output of the fitted KNN model, the best k value is 5 with accuracy 0.9312900; Therefore, the 10-fold cv error rate is 0.06871.



KNN generates more accurate result without estimating the model under linear assumption, but it still cannot give a clear table view of coefficients and tell us which variable is important. More importantly, it suffers from curse of dimensionality. In this problem setting, this issue is more serious because there are around 10 predictors if we include variable Type 1, Type 2 and Generation.

Hence, the KNN method is easier to overfit the model, which leads to a large variance and small bias. Due to these disadvantages, we would try another method called QDA. The QDA model can act as a conciliation to avoid the problem of overfitting while retaining the property of nonlinearity.

3.3 Quadratic Discriminant Analysis

Quadratic discriminant analysis gears classification problem by assuming that the observations within each class are drawn from a multivariate Gaussian distribution with a class specific mean vector and covariance matrix for each class. Hence, the resulting discriminant function is nonlinear and so does the decision boundary.

Including all the quantitative variables: HP, Attack, Defense, Speed Attack, Speed Defense and Speed, we performed 10-fold cross validation to fit the QDA and to predict legendary Pokémon. The resulting cross-validation error rate is 0.0611008, which is lower than that of both logistic regression and KNN method.

By assuming the observations within each class are drawn from a multivariate Gaussian distribution, we were restricted to only include quantitative independent variables. Consequently, we expected QDA to perform worse than the previous two models, as we gave up upon including Type1, Type2 and Generation in the model fitting. However, the resulting cross-validation error rate was improved comparing to KNN method. The QDA model outperforms the Logistic Regression and KNN method, as it serves as a compromise between the non-parametric KNN method and the linear logistic regression approaches. Compared to logistic regression, QDA assumes a quadratic decision boundary which approaching closer to our goal. Different from KNN which suffers from the curse of dimensionality, QDA sets assumption on the decision boundary and prevents it from overfitting the data. To sum up, the QDA method is more preferred to predict the legendary Pokémon.

We used a QDA method with a 10 cross-fold CV error of 0.0611008 to classify whether a pokemon is legendary as our answers to question2.

4. Summary and Discussion

We compared random forest model and boosted tree model to classify the Type 1 of Pokémon and chose boosted tree model as the final model to classify. The important variables to define the type 1 of Pokémon are:
Sp..Atk>Total>Attack>Defense>Speed>HP>Sp..Def>Generation>Legendary.

As an exploration, we added a feature called combination of Type1 and Type2 to define the type of Pokémon. We used a decision tree method to explore which variable can define this new feature of type 1. We recognized that the generation, defense and attack speed can define the combination types of the Pokémon.

After comparing three models in part two, we decided to use QDA as a preferred model to predict whether a Pokémon is legendary or not. Although the 10-fold CV error rate does not vary too much among three models, logistic regression and KNN both have limitations in this problem setting. For future research and improvement in the second model, we can try to add ROC curve as a metric to test the performance of the KNN, logistic and QDA method.

5. R Code Appendix

Data Wrangling
```{r}
# Load in the data
Pokemon=read.csv(file="E:/Xu/Cornell/STSCI4740_SL/Project/Pokemon.csv",head=TRUE,na.strings=c(
""))
library(caret)
# Get the general information of the dataframe
str(Pokemon)
attach(Pokemon)
```

EDA
```{r}
# Missing Value Check
# Check for the missing values and  look how many unique values there are for each variable using the
sapply()
dim(Pokemon)
sapply(Pokemon,function(x) sum(is.na(x)))
# Summarize the unique values
sapply(Pokemon, function(x) length(unique(x)))
# Make a visual take on the missing values
library(Amelia)
missmap(Pokemon, main = "Missing values vs observed")
# Deal with the missing value
Pokemon$Type.1[is.na(Pokemon$Type.1)]="Poison"
# Color the different labels using plots
Pokemon$Generation=factor(Pokemon$Generation)
Pokemon$Legendary=factor(Pokemon$Legendary)
Pokemon$Type.2=as.character(Pokemon$Type.2)

```

```{r}
## Data Visualization
sapply(c("ggplot2","gridExtra","ggExtra","dplyr"), library, character.only = T, logical.return = T, quietly
= T, warn.conflicts = F)
colors <- c("#593420","#080c26","#7d8228",
"#e2dc2f","#fcdefb","#662714","#c91f01","#d0edef","#7f7682","#2d6633","#5b5550","#8dd9f4","#e0e0
e0","#126347","#4a116b","#332f28","#bababa","#203bc1")

## Check the distribution of Type.1 Pokemon
ggplot(Pokemon,aes(x=Type.1,fill=Type.1))+
```

```
  geom_histogram(stat="count",color="black")+
  scale_fill_manual(values=as.character(colors))+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+labs(title="Number of Pokemon per
Type.1",x="Type.1",y="Count")+theme_light()
```
```{r}
# check the distribution of Type.2 Pokemon
ggplot(Pokemon,aes(x=Type.2,fill=Type.2))+
  geom_histogram(stat="count",color="black")+
  scale_fill_manual(values=as.character(colors))+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+labs(title="Number of Pokemon per
Type.2",x="Type.2",y="Count")+theme_light()
```

Conclusion drawn from the EDA:
Nearly half of the Pokemon have only one primary type.
Flying type as the first type is very rare.
The water and normal are the most common primary type.

Question 1:Classification for Types
Question1.1 Classification for only one type

```{r}
# Fit a random forest model to ask question 1

# Using Caret Method
# A crucial step to make caret library work
Pokemon$Legendary <- factor(Pokemon$Legendary, labels = c("yes", "no"))
Legendary=factor(Pokemon$Legendary)
# Set a fitcontrol model
fitControl <- trainControl(method = "cv",number = 10)
# make sure no empty classes in y
Pokemon$Type.1[is.na(Pokemon$Type.1)]="Poison"
RF1=train(Pokemon[,-(1:4)],Pokemon[,3],method="rf",trControl = fitControl,verbose = FALSE)
print(RF1)
varImp(RF1)
plot(RF1)
# boosted tree method
gbmFit1=train(Pokemon[,-(1:4)],Pokemon[,3],method="gbm",trControl = fitControl,verbose = FALSE)
print(gbmFit1)
varImp(gbmFit1)
plot(gbmFit1)
```

Check the combination of Types
Collinearity Check
```{r}
library(GGally)
```

```
ggpairs(Pokemon[,6:11])
```

```{r}
# copy a dataframe
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
library(party)
library(partykit)
library(caret)
pokemon.copy=read.csv(file="E:/Xu/Cornell/STSCI4740_SL/Project/Pokemoncopy.csv",head=TRUE,na.strings=c(""))
# add a new columnn
library(tree)
form=as.formula(pokemon.copy$DualType~Total+HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Generation)
tree.1 <- rpart(form,data=pokemon.copy,control=rpart.control(minsplit=20,cp=0))
prp(tree.1,varlen=15)
```

Question1.2 Classification for two types
Dual Type Analaysis Function

Question2:BUilding Predicting Models
```{r}

# Fit a logistic regression model on the Legendary Variable to answer the question 2
logmodel <-
glm(Legendary~Defense+Sp..Atk+Sp..Def+Speed+Generation,family=binomial,data=Pokemon)
summary(logmodel)
# Interpret the results of logistic regression model
# First of all the Attack variable is not statistically significant
# All the coefficients are positive,which means the higher the attributes, the more likely a poker man is going to be a legendary
anova(logmodel,test="Chisq")
library(boot)
cost <- function(r,pi =0) mean(abs(r-pi) > 0.5)
cv.err=cv.glm(Pokemon,logmodel,cost,K=10)
cv.err$delta

## Use Caret Library to Try Different Models
library(caret)
# A crucial step to make caret library work
# Set a fitcontrol model
fitControlknn <- trainControl(method = "cv",number = 10)
```

```{r}
# Try KNN models to classify legendary
Pokemon$Type.2[is.na(Pokemon$Type.2)]="no"
knnFit <-
train(Legendary~Type.1+Type.2+Total+HP+Attack+Defense+Sp..Atk+Sp..Def+Speed+Generation,
data=Pokemon,method = "knn", trControl = fitControlknn, preProcess = c("center","scale"), tuneLength =
20)
knnFit
plot(knnFit)
```

```{r}
# Try QDA
fitControl <- trainControl(method = "cv",number = 10)
QDAFit <- train(Legendary~HP+Attack+Defense+Sp..Atk+Sp..Def+Speed, data=Pokemon,
method='qda',trControl=fitControl)
QDAFit

```