Dovie Solomon

Professor Kovtun

BICI

Final Project

**Introduction/Description:** My data set was taken from dataworld and consists of the different Major Tennis Tournaments of the 2011 season, from both male and female athletes. During each match there are many different statistics that show up, like Winners, Aces, Net Points, the columns of the data set comprised of different statistics from each match played between the two players. For example, TPW.1 is total points from player 1 while TPW.2 is the same for player 2. I added 2 columns of my own which showed what style ground the game was played on and what the gender of the athletes was. I also had to make sure to get rid of the statistics that meant a player won, like Total Points, Set Points, and some that are not definitively showing a winner but to my knowledge ensures a winner almost every time (Break Points Won). My goal was to predict who the winner of the match was given different these remaining different statistics. I had built many different models to see what performed the best under a 4 cross-validation. My elaboration on each model and their effectiveness is listed below.

**Methodology:** During the process to determine the best model, I used the 4 cross-validation. This allowed me to test the model on an unknown part of the predicter to maximize its similarity to predicting the future. After testing each model, I calculated the misclassification rate which is the difference between what actually happened and what I predicted. We want a low misclassification rate and the model with the lowest is the one I suggest using. I have used two different styles as you will see below: Linear Classifiers (Logistic) and Non-Linear (KNN, SVM, Decision Trees, Etc.). I also wanted to note that I used a different script for each model so that I could keep track and reset different variables per the needs of each model. (As you will see, some need Standardizing, removal of qualitative, or removal of certain variables)

Logistic (3 Models):

The idea behind a Logistic model is that each predictor variable has a weight to equal a “y” variable. With the help of Algebra, we determine that the probability of an event occurring can either be set to above 50% or below that and after predicting each column we then use this transformation. To determine the misclassification of the model I used a 4-Fold CV with the aid of AIC. The lowerthe AIC the better. First, I used the backward method which calculates all the AIC by removing 1 variable at a time. If there are any results with a lower AIC then it removes the variable with the lowest and continues this process until there is no lower AIC. I also used the Forward method, which is similar in that it starts with zero variables and adds until it is not possible to create a lower AIC than the previous additions. The last model in this section is called Principle Component Analysis Model (PCA). This is where we do not remove correlated variables as the PCA will neutralize them for us. We then use a plot to determine how many PCA we should use in the model equation to predict why, just as we did before.

Decision Trees (3 Models):

This style of model allows for qualitative variables so I have left those in. A decision Tree model allows a user to follow a branch on the tree plot based on different yes/no characteristics and end up at a prediction. I used 3 different Control Parameters, to create 3 different trees. These different Control Parameters tell the model if it is “allowed” to continue to keep splitting into more branches.

SVM – Support Vector Machine (4 Models):

In this model we can only use quantitative data so I started off my removing those that were qualitative. Then I made sure to standardize the data so that when the model assumes normality it actually is normal. (A distance in one column of 67 to 70 may be the same as one from 1 to 15 on another). An SVM Model is one that separates the class of the y variable by a margin. Each prediction that is on the wrong side has a cost. In my models I set the cost to different amounts in hope to lower the misclassification rate. The dimensions can also be 2D or 3D for this model so I made sure to test it on both.

KNN - K – Nearest Neighbors (2 Models, 50 Neighbors):

Once again for this model I made sure to start off by standardizing and removing the qualitative variables. This is because the model relies upon distance and one can no compute the distance between a man and a woman. The standardizing is to make sure the distances between two numbers in 1 column is the same as ones in another. The idea behind this model is that it will test a certain number of neighbors based on how many I tell it to. During my model testing I used 50 different neighbors in multiples of 5 from 1 to 50. I then found the misclassification rate based again on 4 cross-validation on each one of these neighbors. The lowest misclassification was the model in the KNN which I deemed performed best. If any two were the same or almost the same I chose the lowest neighbors of them as it is most likely to work on a future data.

Cluster Analysis (4 Models):

This model also had to be standardized, for reasons see above. The next part of the cluster method is to divided the data into many groups, in our case it was two. I then had the option to use different style distances, I used both the Manhattan and Euclidian. Next, for each distance style I used single-linkage, which uses distance between the closest clusters, and I also used average-linkage which uses the average distance of all the observations within each cluster.

Random Forest (1 Model):

This model works in a way that it forms a creation of many decision trees where each tree is constructed with randomly selected observations from the original dataset and a subset of variables is selected at each node to figure out the best variable to use. Once all of the original variables have been used or a node has 100% of one class in it, the tree stops. After the creation of many trees, each tree votes for the class of an observation, and the predicted class is the class that gets the most votes from the trees.

4 Cross-Fold Validation:

As mentioned above is some of the paragraphs, I made sure to 4 Cross-Fold Validate for almost each model that I had run to find the misclassification rate. The one with the lowest is the one that I feel will perform best on future data. The reason for the 4-Fold is to best simulate predicting on a set of data outside of the prediction data set, but it still will allow us top check with the actual data to determine how well it does. For those familiar with the train and test set idea, this is very similar except with 4 divisions as opposed to 2.

**Analysis:**

Intro:

The Response variable for my models is Result, which contained 1s and 0s to show the player who won, with 1 being the first player and 0 being the second. My data consisted of 31 predictor variables; 2 with the name of the players, 1 which said the round, 26 with actual statistics (13 for each player), and 2 that I created and added saying if they are Male or if the Court was Hard or Soft. While looking at the statistics I realized that a few were directly showing who won, like TPW (total points won), BPW (break points won), and TSW (total sets won). I removed each of these as well as the Player names variable from the data set so that it could more accurately predict the winner based on unique statistics. Before beginning any sort of other Model, I made sure to get rid of any columns with NA. I made sure to run the code of this intro before running any new model.

Logistic:

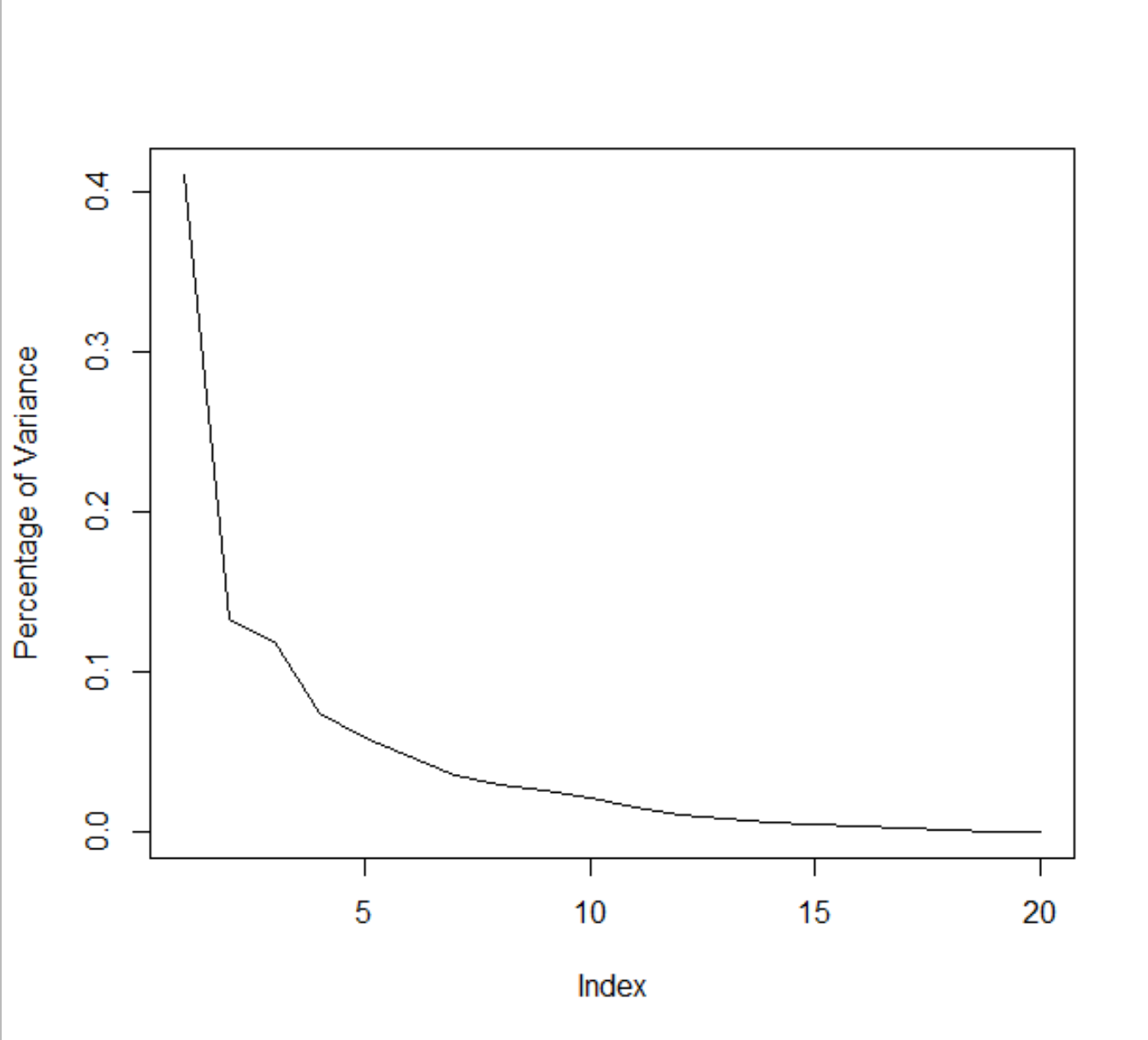
This category had three models, the first model I had I made sure to use Result, which is 1s and 0s as my Y variable. After generating a model, I had to find correlated data and make sure that none of my predictor variables were correlate with each other. To do that I have generated a heatmap which I have provided below. After running the heatmap and removing variables a few times I ended up with 8 predictor variables and 1 Y variable. Using the Step function and a loop I was able to determine which variable to use out of these 8 would perform best. This method uses AIC as a cutoff and performs via backward selection. My misclass rate was **17.76%** for this process using the 4-Fold CV method too. The predictor variables it said to use were, ACE.1, ACE.2, FSP.1, FSP.2, SSW.1, SSW.2. The equation came out to be is that the probability of y=1 = p =

The next model was also based off of AIC but I used Forward selection. This under the 4-Fold CV method, resulted in a misclassification of **19.83%,** but provided us with the same variables as above. The next model I used was a PCA model with all the possible variables. I removed the qualitative and response variable, then created the PCA variables. I then found the misclassification with the 4-Fold CV method which came out to **43.38%.**  The equation for the model is

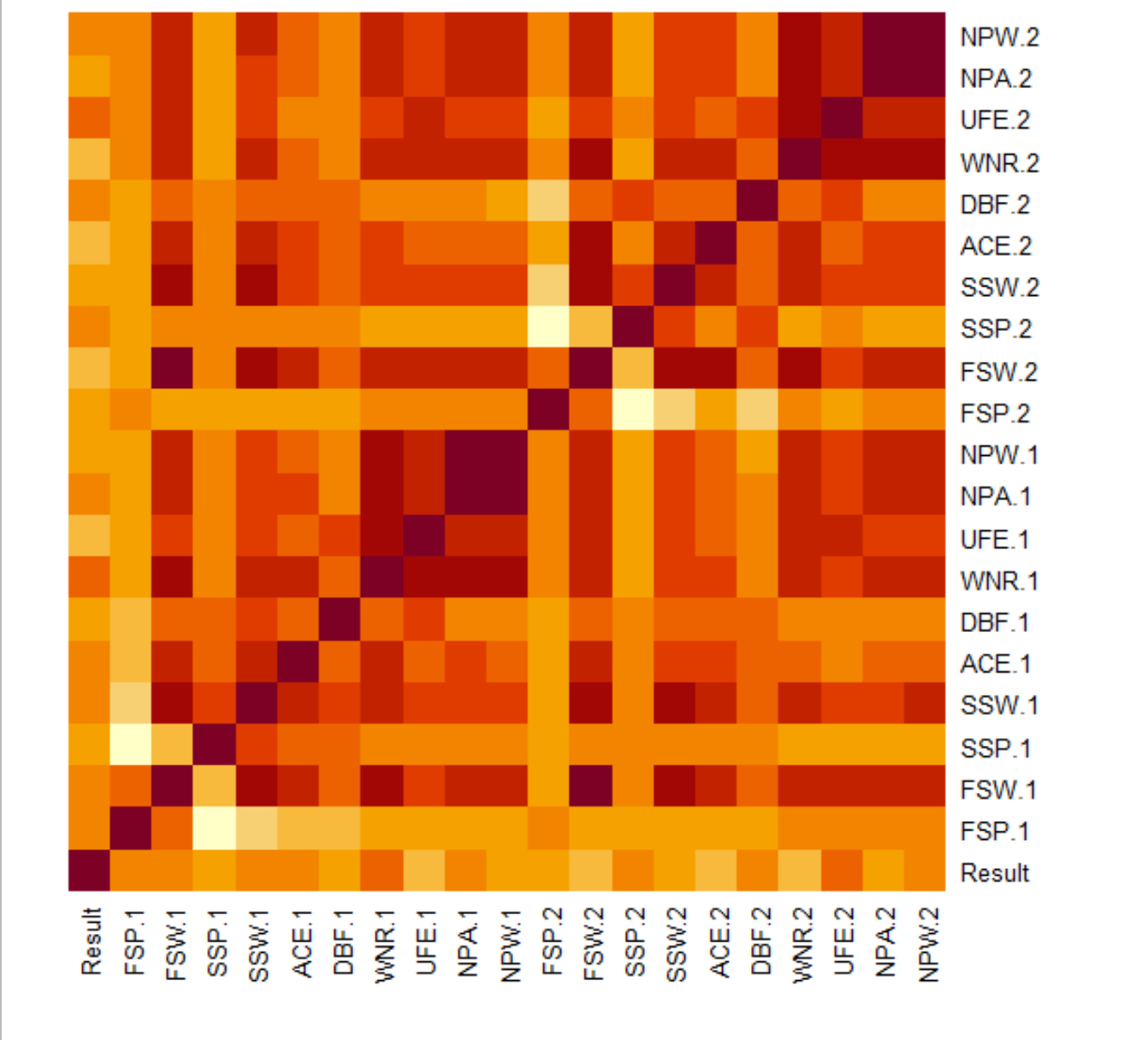
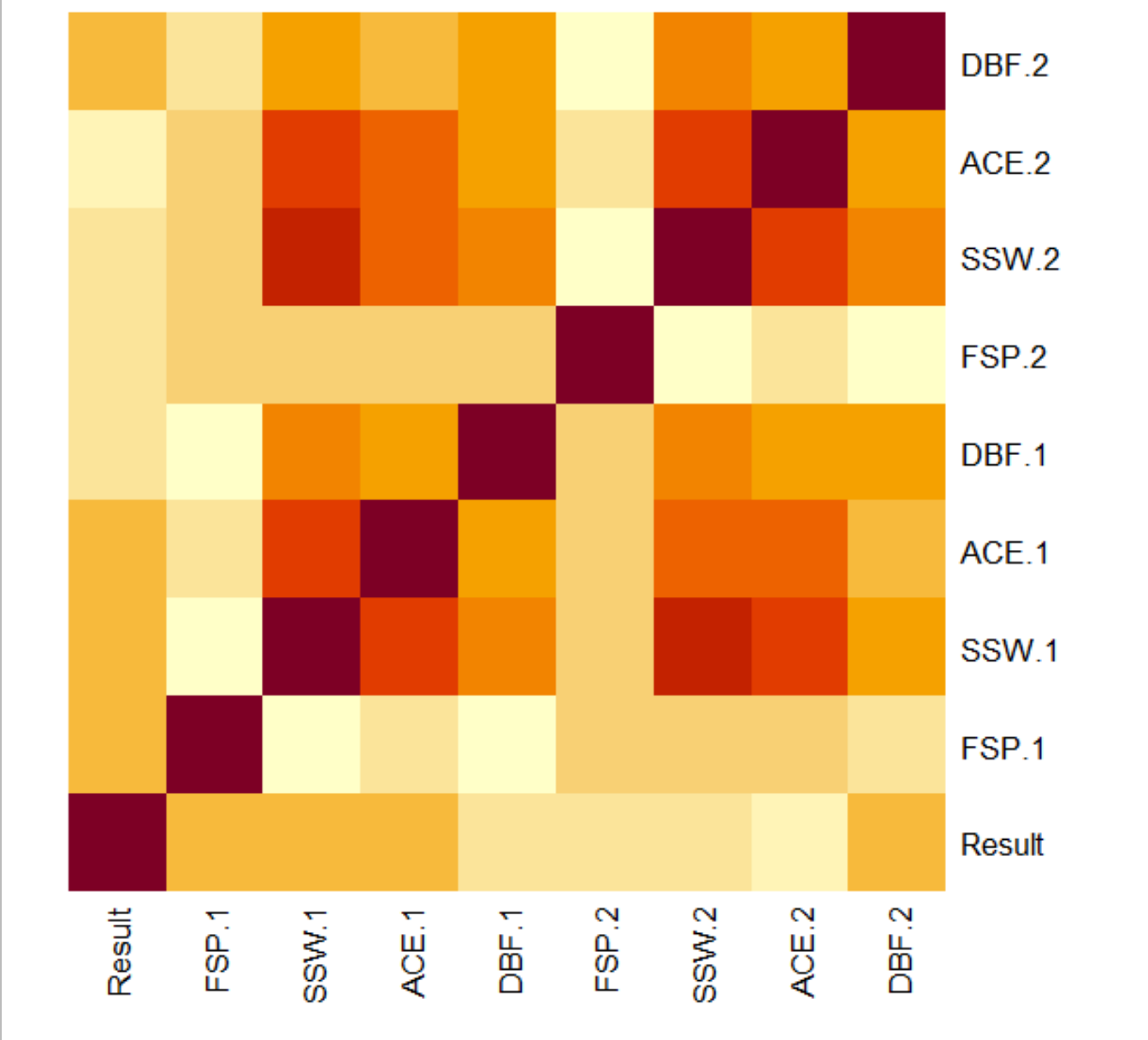
**The best model is the Backward Selection with AIC with a misclass of 17.76%**

Plots for the PCA and Heatmaps:

PCA Plot:



Heatmap Before: Heatmap After:

Decision Trees:

The first Model I tested in this category was with a control parameter set to the default, which is .01. The Y variable I used here was a character version of the 1 and 0s I wrote about above. This is because we don’t want it to be numeric (And I am not a fan of Factors). I then proceeded to set up my 4-Fold CV as presented above. After running my loop with the default parameter, the misclassification rate came out to **29.13%.** Without the CV, the misclassification for this Model was **12.52%.** I was provided the Tree Plot below as Image 1. The next Model I ran was with a control parameter of .015. This is because it is based on the ideal X-Error. I have also provided the cp plot below as Image 2. The misclass with this CP under the 4-Fold CV was **30.09%,** while without the CV it was **14.66%.** The last Model I used had a CP of .05. The misclass under this with the 4-Fold CV was **39.25%** and without CV it was **25.46%.** I have provided a image of the Tree under Image 4.

**The best model under the 4-Fold CV is the default CP with a misclass of 29.13%. The Logistic Model Backward Selection is still the best model.**

Image 1:

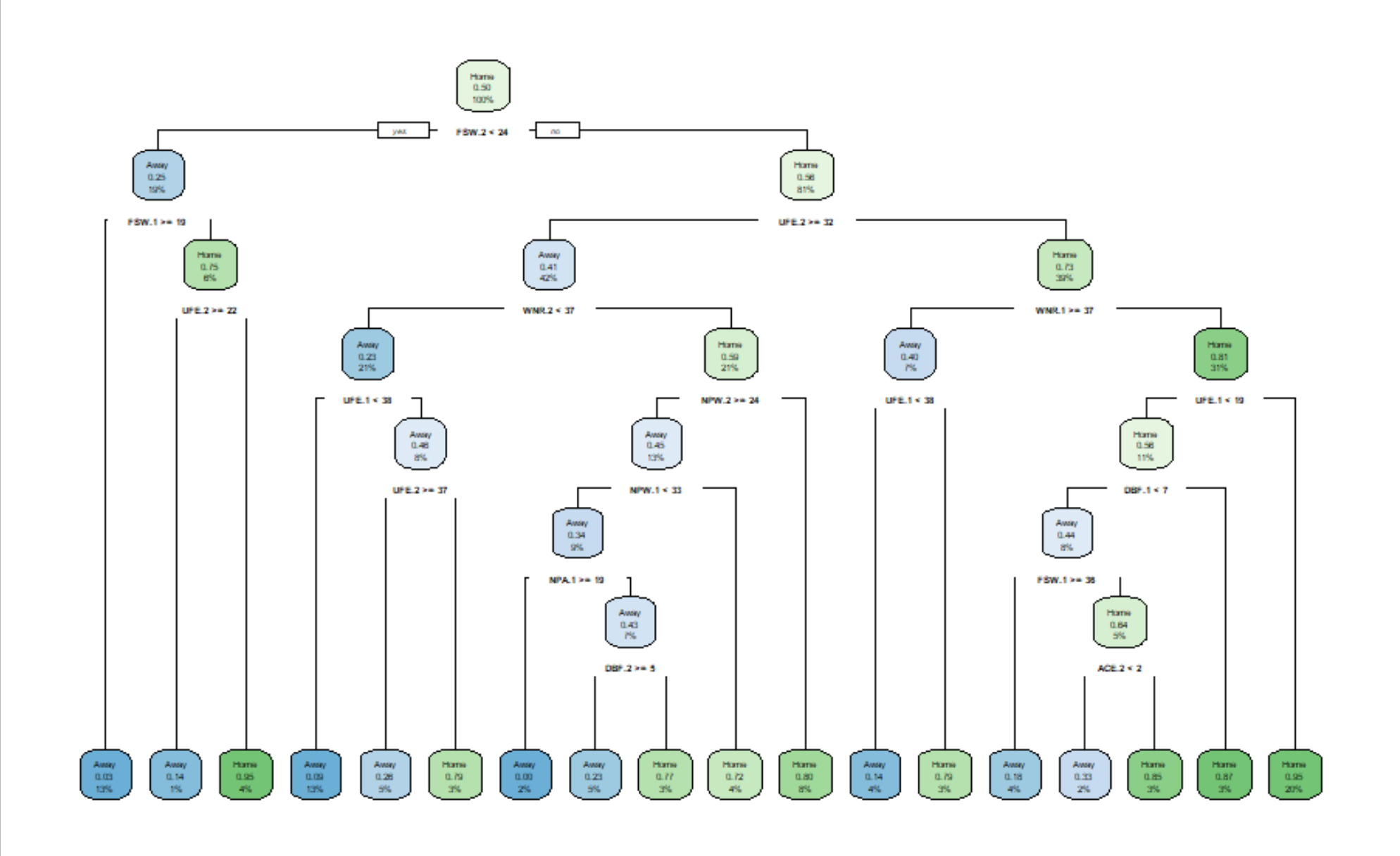


Image 2:

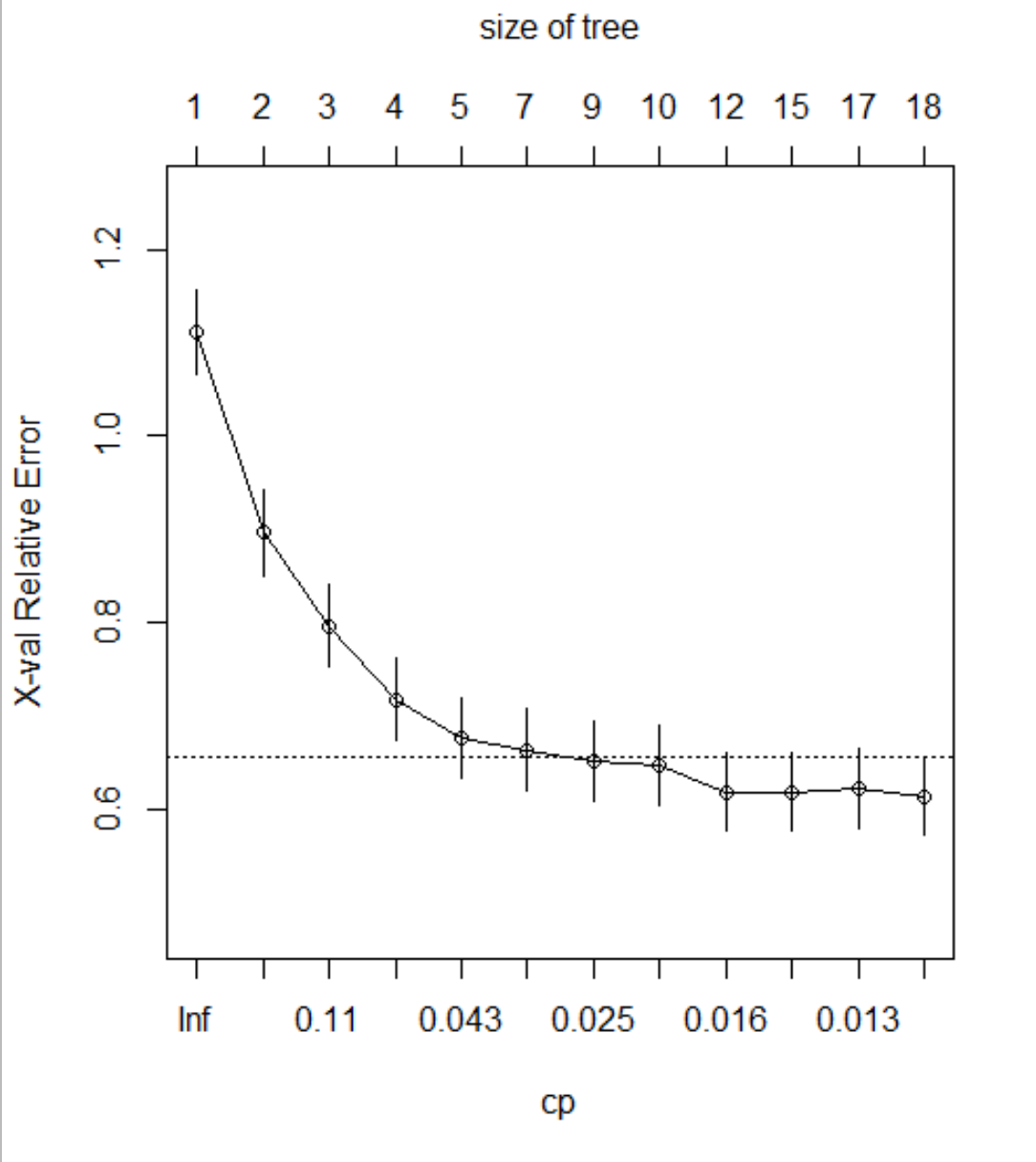


Image 3:

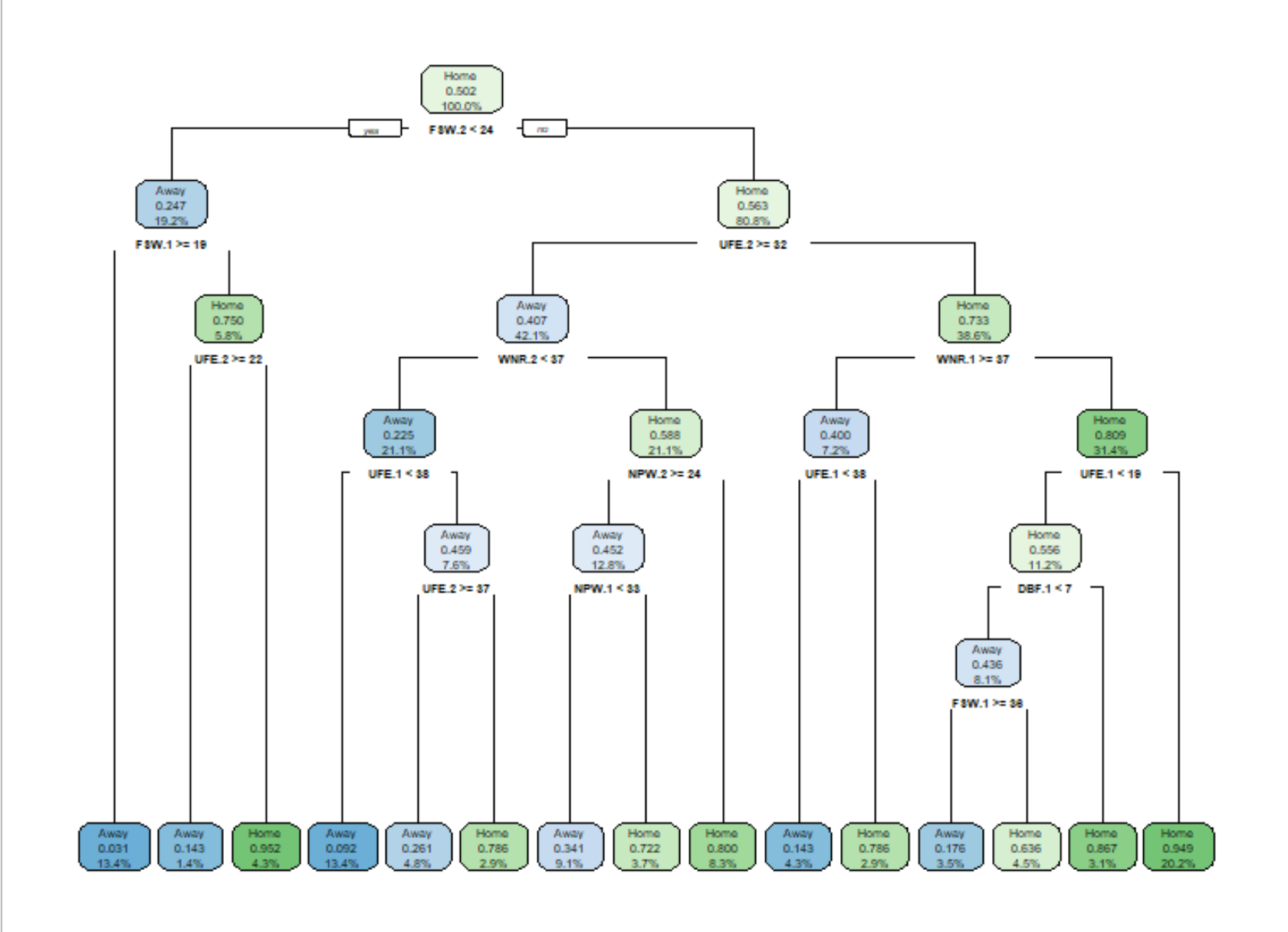
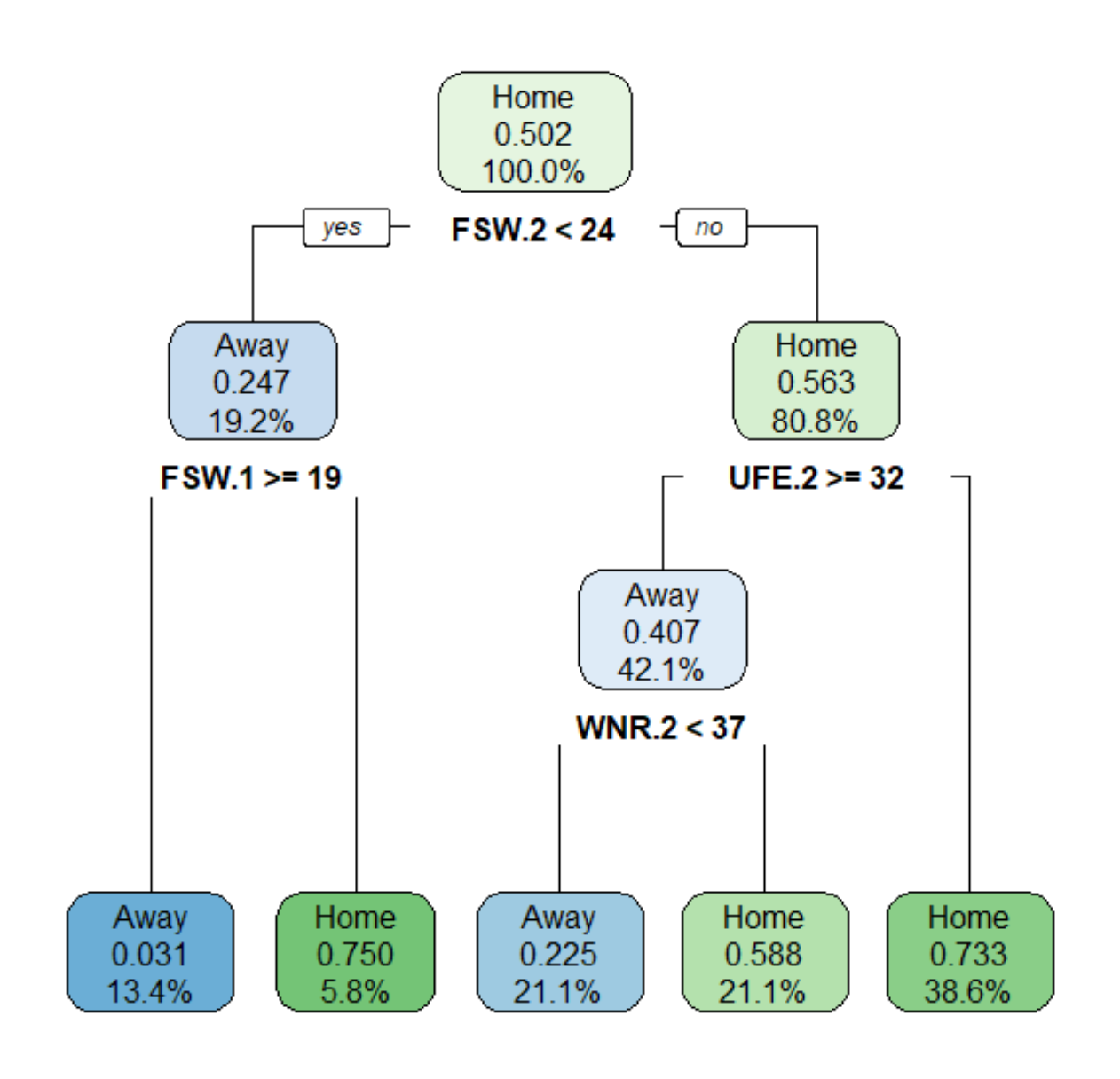


Image 4:



Support Vector Machine (SVM):

The first model I ran for this category was with a Kernel set to Radial and a Cost of 1. The Variables I chose to use in this model were the same ones used in the Logistic by eliminating using a heatmap. They are FSP.1, FSP.2, SSW.1, SSW.2, ACE.1, ACE.2, DBF.2, and DBF.1 with the response variable Who Won which is a factor of Home and Away. The resulting misclass under 4-Fold CV was **25.82%.** I then ran the model with a Kernel still set to Radial but a Cost set to 100. The misclass under the 4-Fold CV for this model was **28.30%.** These models are 3 Dimensional and therefore do not have a plot. The next model I ran was with a Kernel now set to Linear and a Cost back to 1. The resulting misclass was **25.82%** as well. Lastly, I ran the model with a Kernel set to Linear and the Cost at 100. The misclass rate was **28.30%.** There was no difference between Radial and Linear only between Costs. I have provided 4 plots below. Image 1 and 2 are for a Cost of 1, and 3 and 4 are for the Cost of 100. The x and y axis can be seen in the plot themselves.

**The best model under the 4-Fold CV is the Radial or Linear with a Cost of 1 resulting in a misclassification of 25.82%. The Logistic Backward Selection Model is still the best.**

Image 1:

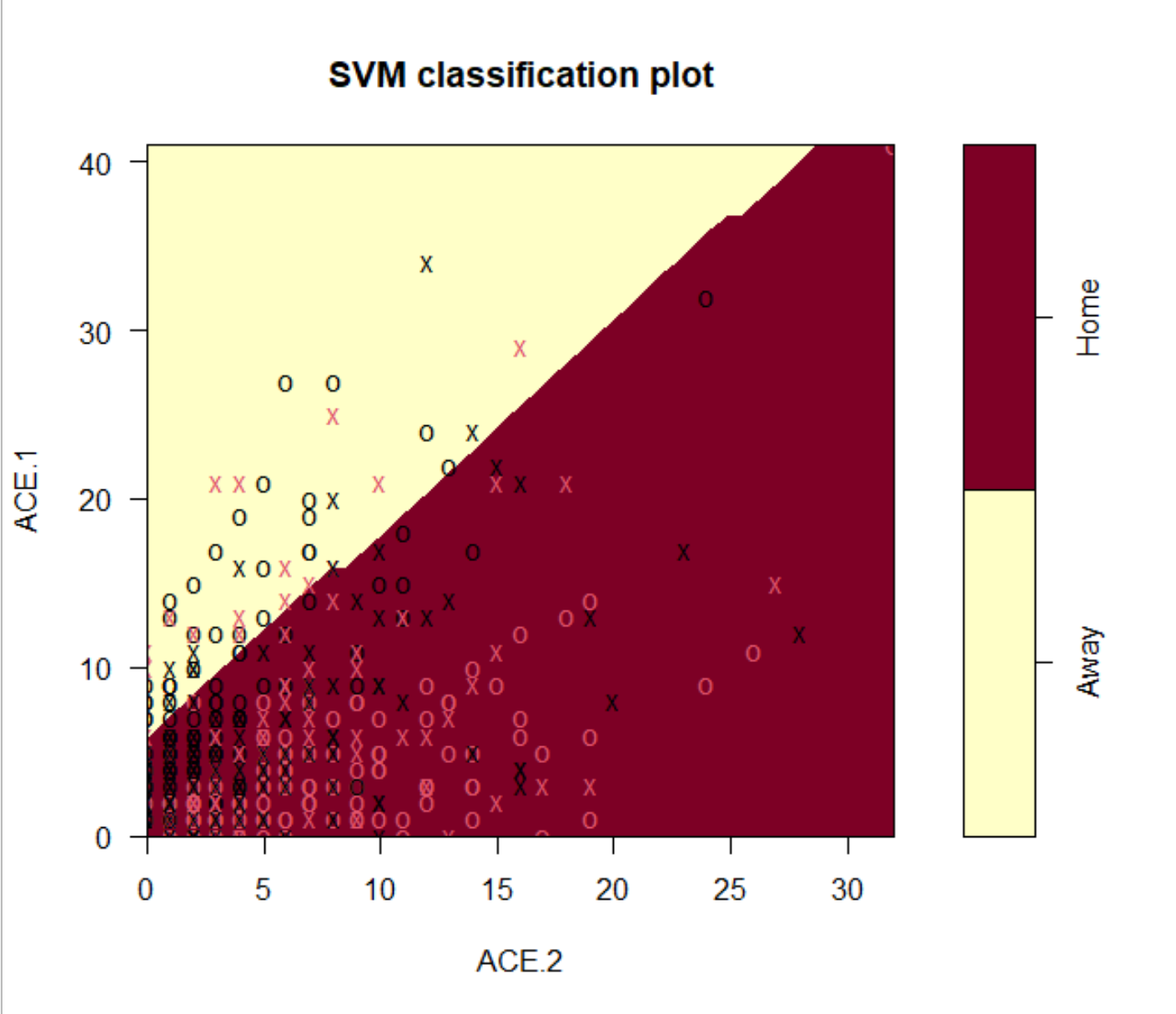


Image 2:

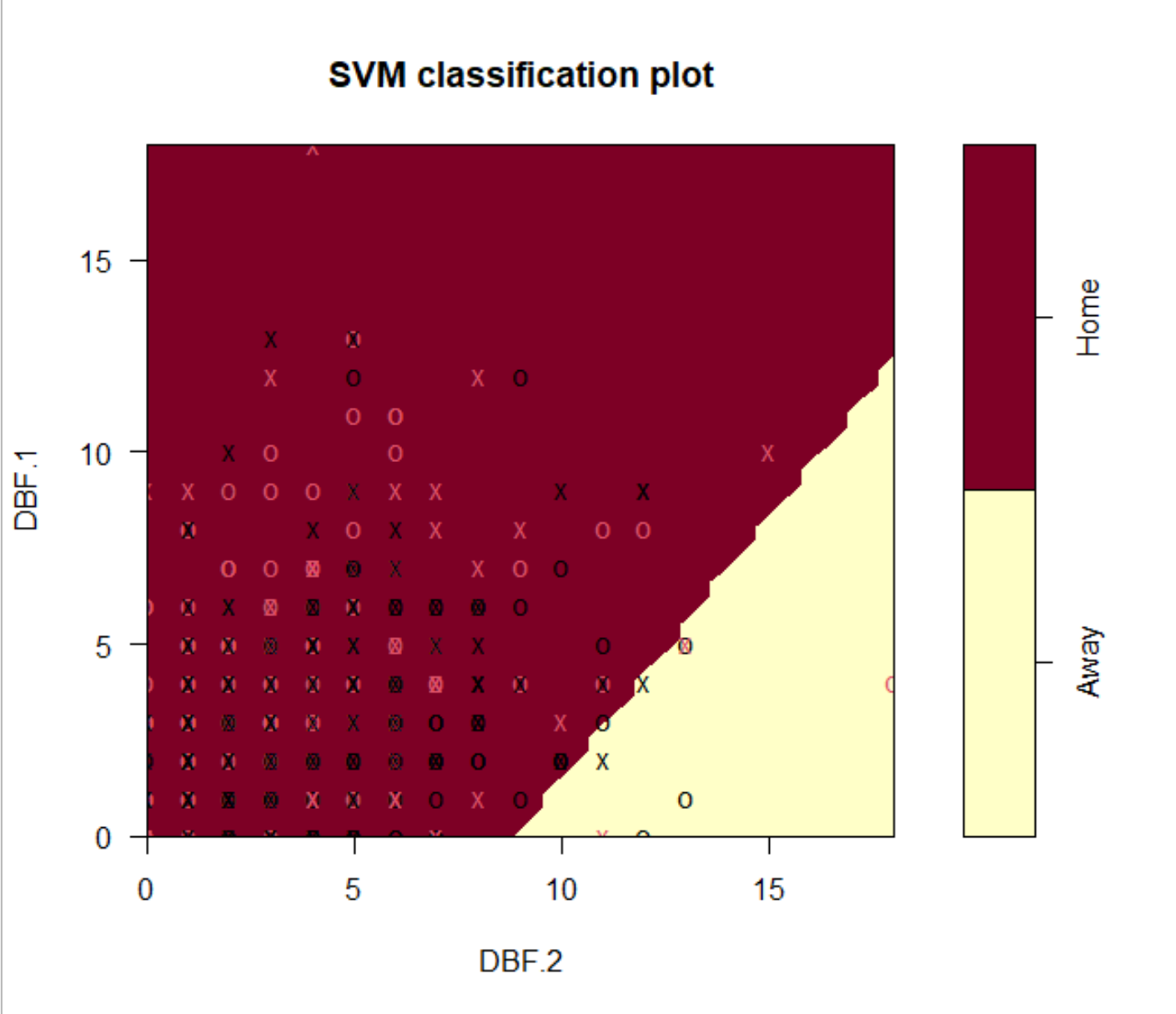


Image 3:

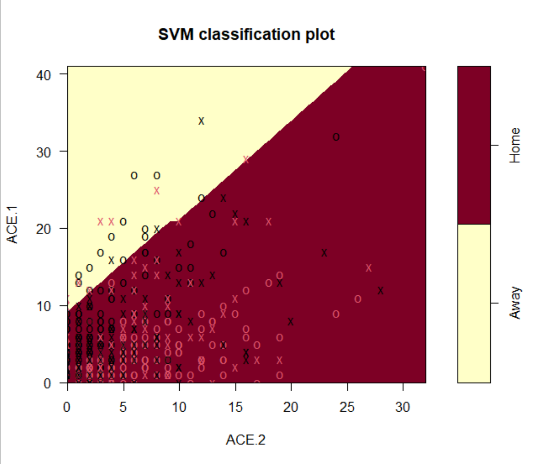
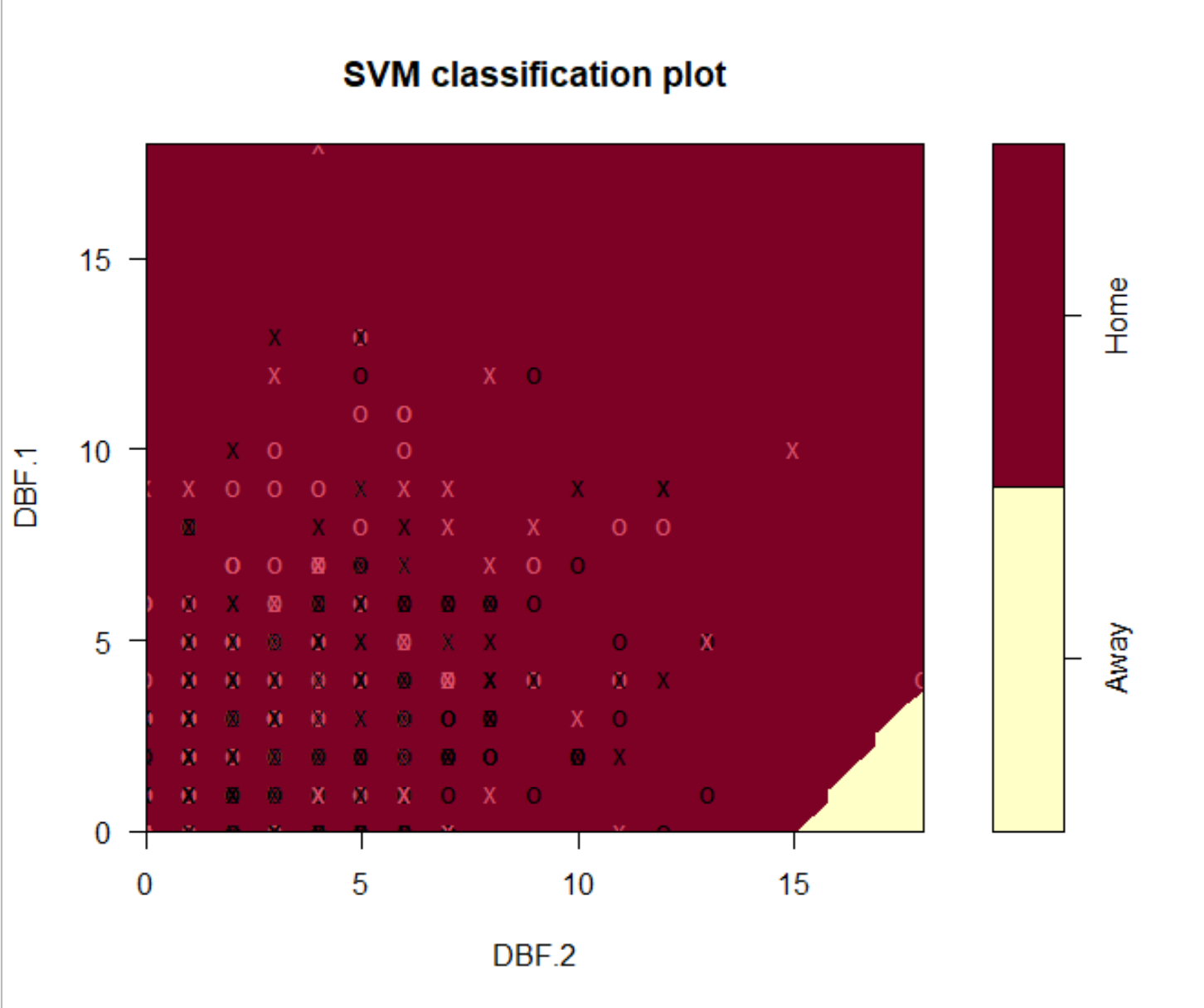


Image 4:

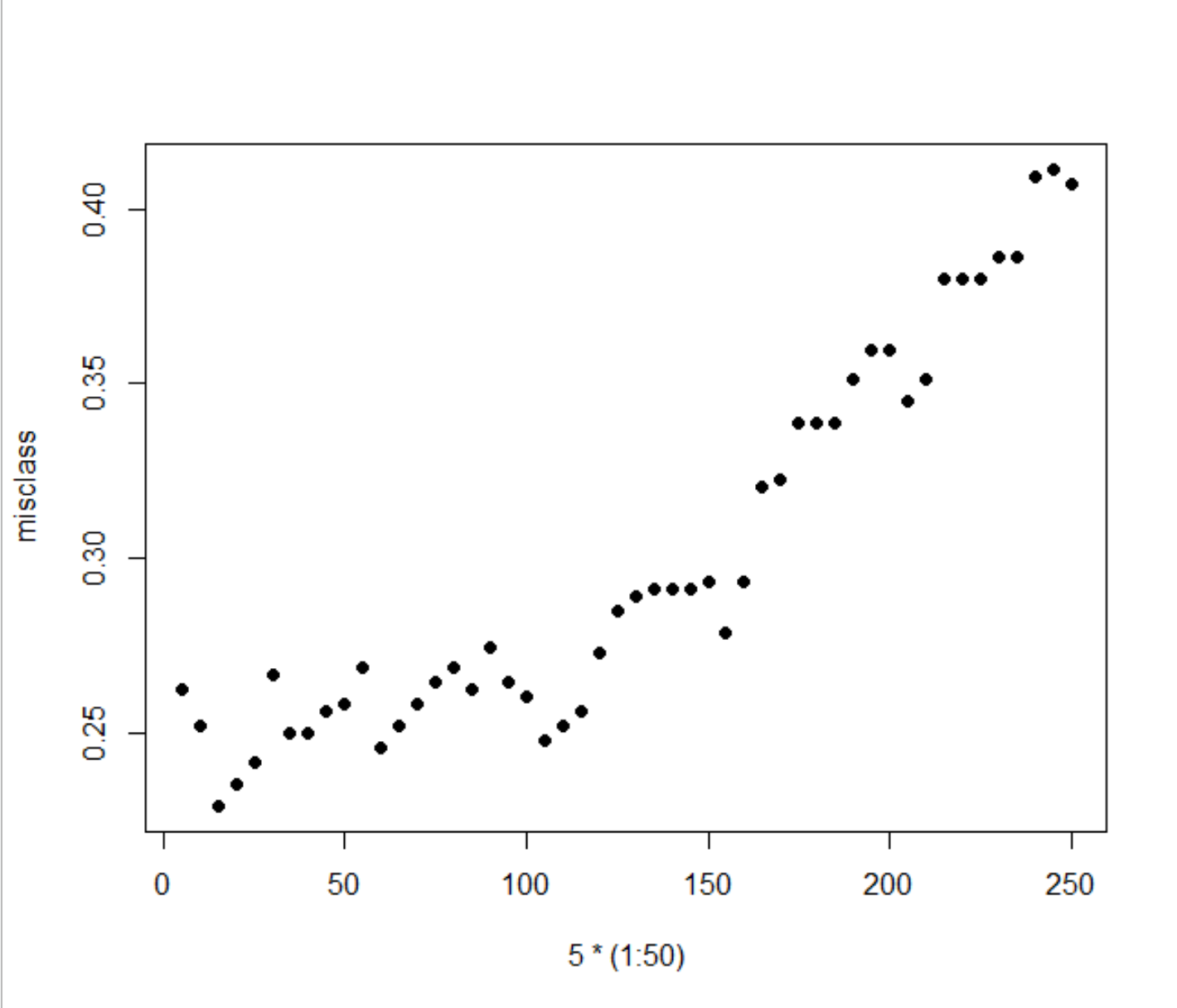
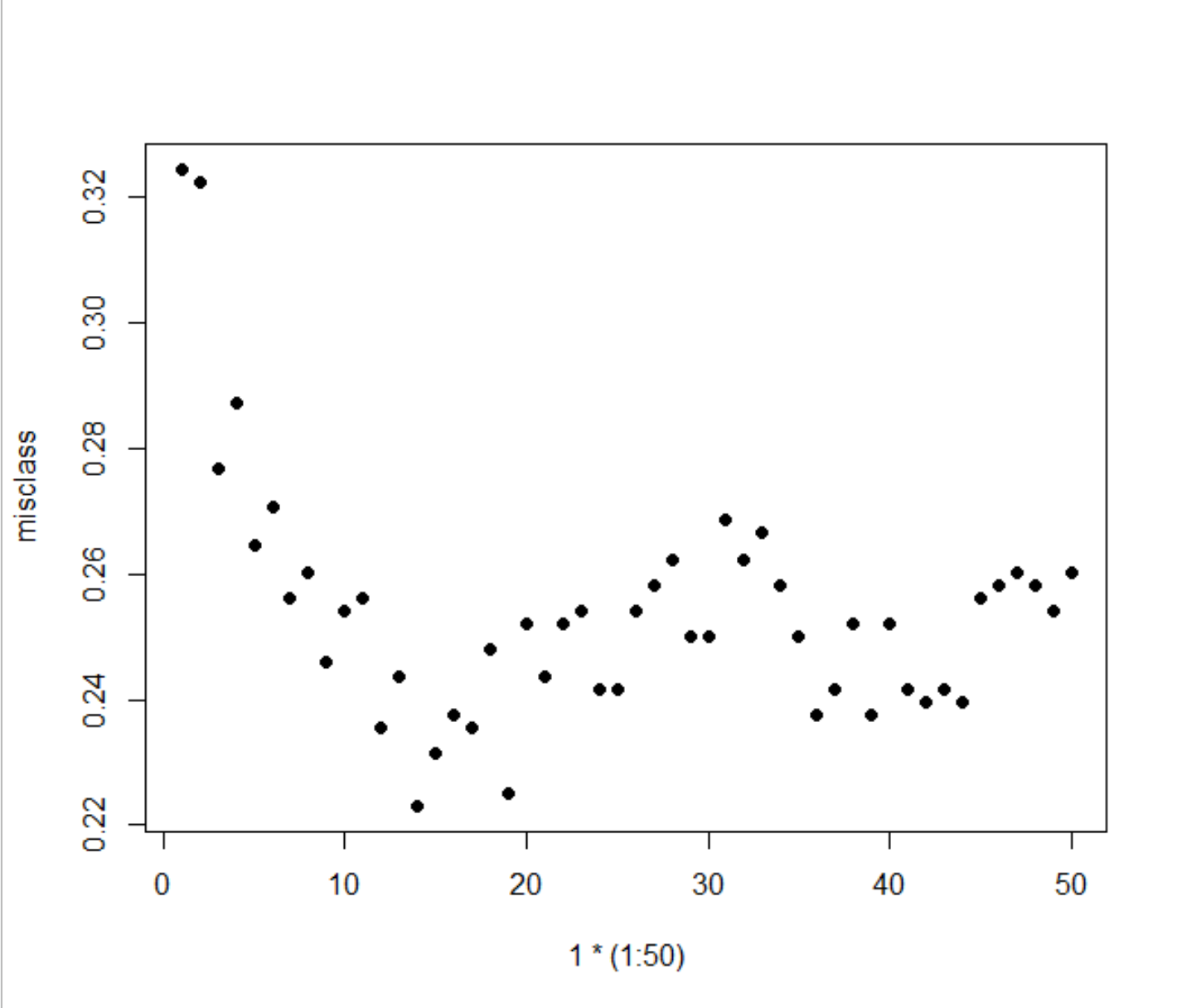


K- Nearest Neighbors (KNN):

The first thing I did under this model was to standardize the columns as stated above because it relies upon distance. The Y variable I used here was once again the binary version called Results. To start I created a loop that would calculate the misclass rate based of a 4-Fold CV using 50 of the nearest neighbors by every 5th neighbors (5, 10, 15….). The lowest misclass was **22.93%.** I then plotted all the misclasses to give us a better view of the trends. I also realized that since this misclass was inside the first 50 neighbors I ran the model again; this time going by ones and not fives. The lowest misclass now was **22.31%,** lower than before, which came out to be the 14th neighbor. I once again plotted the misclasses to see the trends.

**The best model under the 4-Fold CV is the 14th Neighbor which had a misclassification of 22.31%. The Logistic Backward Selection Model is still the best so far.**

Image 1: Image 2:

Clustering:

The first thing I did here was to make sure that the Data was standardized just like the KNN. I then used all the quantitative data. Each model I ran I had them divided into 2 Clusters. After that I ran a model with Average-Linkage and using the Manhattan Distance. Group 1 came out to be Away and Group 2 being Home. The misclass rate for this model came out to be **49.58%.** The next model I ran was Complete-Linkage with the Manhattan Distance. Group 1 once again was Home and Group 2 was Away. The misclass rate for this model came out to be **49.79%.** The next model I ran was Average -Linkage with the Euclidian Distance. Group 1 was predicted to be Home and Group 2 was Away. The misclass rate for this model came out to be **50%.** The next model I ran was Complete-Linkage with the Euclidian Distance. Group 1 once again was Home and Group 2 was Away. The misclass rate for this model came out to be **47.32%.**

**The best model under this method is Euclidian Distance with Complete Linkage which had a misclassification of 47.32%. The best model is still the Logistic Backward Selection.**

Random Forests:

Before running this model, I made sure to convert my Who\_Won variable from a character to a factor. For this model, I created a classification type Random Forest model with 1000 trees and a mtry of .5\*10. This model had a misclass rate of **17.92%.**

**This is model of 1000 trees has a misclassification of 17.92%. It is close but Logistic Backward Selection is still the better model.**

**Conclusion:**

After this long process the best model to use under these circumstances to predict the y variable of “Who is the Winner”, the best model to use is the Stepwise Regression Model with AIC Backward Selection. This is because testing it under 4-Fold Cross Validation, the Misclassification rate of the predictions is **17.76%.** 4-Fold is important as it will show the most realistic to predicting things outside the data, while still using all the data provided. So while other models may perform better outside 4 Fold CV, it is my suggestion to use this model to predict future winner and losers.

**Appendix:**

Intro: - This was run before each model as a reset since many models used different stuff.

Creating New Variables and Combining:

M\_or\_F <- rep("Female",127)

Hard\_or\_Soft <- rep("Hard",127)

AusWomen <- cbind(AusOpen.women.2013,M\_or\_F,Hard\_or\_Soft)

AusWomen <- AusWomen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Male",126)

Hard\_or\_Soft <- rep("Hard",126)

AusMen <- cbind(AusOpen.men.2013,M\_or\_F,Hard\_or\_Soft)

AusMen <- AusMen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Female",127)

Hard\_or\_Soft <- rep("Soft",127)

FrenchWomen <- cbind(FrenchOpen.women.2013,M\_or\_F,Hard\_or\_Soft)

FrenchWomen <- FrenchWomen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Male",125)

Hard\_or\_Soft <- rep("Soft",125)

FrenchMen <- cbind(FrenchOpen.men.2013,M\_or\_F,Hard\_or\_Soft)

FrenchMen <- FrenchMen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Female",76)

Hard\_or\_Soft <- rep("Hard",76)

UsWomen <- cbind(USOpen.women.2013,M\_or\_F,Hard\_or\_Soft)

UsWomen <- UsWomen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Male",126)

Hard\_or\_Soft <- rep("Hard",126)

UsMen <- cbind(USOpen.men.2013,M\_or\_F,Hard\_or\_Soft)

UsMen <- UsMen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Female",122)

Hard\_or\_Soft <- rep("Soft",122)

WimbWomen <- cbind(Wimbledon.women.2013,M\_or\_F,Hard\_or\_Soft)

WimbWomen <- WimbWomen[,c(-(20:24),c(-(38:42)))]

M\_or\_F <- rep("Male",114)

Hard\_or\_Soft <- rep("Soft",114)

WimbMen <- cbind(Wimbledon.men.2013,M\_or\_F,Hard\_or\_Soft)

WimbMen <- WimbMen[,c(-(20:24),c(-(38:42)))]

colnames(UsWomen)<-colnames(AusMen)

colnames(UsMen)<-colnames(AusMen)

colnames(WimbWomen)<-colnames(AusMen)

colnames(WimbMen)<-colnames(AusMen)

colnames(FrenchWomen)<-colnames(AusMen)

colnames(FrenchMen)<-colnames(AusMen)

TennisDataAll <- rbind(AusMen,AusWomen,UsMen,UsWomen,WimbMen,WimbWomen,FrenchMen,FrenchWomen)

Omiting the NA and Removing “Bad Variables”

TennisDataAll=na.omit(TennisDataAll)

TennisDataAll <- TennisDataAll[,c(-1,-2,-3,-5,-6,-15,-16,-19,-28,-29,-32,-33,-34,-35)]

Logistic:

Creating Heatmap:

model=glm(Result~., family=binomial(),data=TennisDataAll, control = list(maxit = 50))

model

summary(model)

CorrMatrix <-cor(TennisDataAll)

heatmap(CorrMatrix,Rowv=NA,Colv=NA,scale="none")

CorrMatrix

levelplot(CorrMatrix,scales=list(x=list(rot=90)))

TennisDataAll <- TennisDataAll[,c(-3,-11,-13,-21)]

TennisDataAll <- TennisDataAll[,c(-7,-8,-9,-15,-16,-17)]

TennisDataAll <- TennisDataAll[,c(-3,-8)]

model=glm(Result~., family=binomial(),data=TennisDataAll, control = list(maxit = 50))

model

summary(model)

CorrMatrix <-cor(TennisDataAll)

heatmap(CorrMatrix,Rowv=NA,Colv=NA,scale="none")

CorrMatrix

levelplot(CorrMatrix,scales=list(x=list(rot=90)))

Model 1: - Stepwise with AIC Backward Selection:

TennisDataAll <- TennisDataAll[1:484,]

g1 <- 1:121

g2<- 122:242

g3 <- 243:363

g4 <- 364:484

Groups <- data.frame(g1,g2,g3,g4)

predictions=c()

for(i in 1:4){

model = step(glm(Result~., family=binomial,data=TennisDataAll[Groups[,i],], control = list(maxit = 50)))

Pred = predict(model,newdata=TennisDataAll[-Groups[,i],])

Pred=(fitted(model)>.5)\*1

predictions=c(predictions,(Pred))

}

sum(TennisDataAll$Result!=predictions)/length(predictions)

Finding the Equation:

glm(Result~ ACE.1 + ACE.2 + FSP.1 + FSP.2 + SSW.1 + FSW.2, family=binomial,data=TennisDataAll, control = list(maxit = 50)))

Model 2: - Forward Selection with AIC

predictions=c()

for(i in 1:4){

model = step(glm(Result~1,data=TennisDataAll[Groups[,i],]),direction="forward",scope=list(lower=glm(Result~1,data=TennisDataAll[Groups[,i],]),upper=glm(Result~.,data=TennisDataAll[Groups[,i],])))

Pred = predict(model,newdata=TennisDataAll[-Groups[,i],])

Pred=(fitted(model)>.5)\*1

predictions=c(predictions,(Pred))

}

sum(TennisDataAll$Result!=predictions)/length(predictions)

Model 3: - PCA

Getting the PCA data and the Plot

pca=prcomp(TennisDataAll[,-1,-22,-23],scale=TRUE)

plot(pca$sd^2/sum(pca$sd^2),type="l",ylab="Percentage of Variance")

Getting the Misclass – CV

PCdata=data.frame(pca$x,TennisDataAll$Result)

g1 <- 1:121

g2<- 122:242

g3 <- 243:363

g4 <- 364:484

Groups <- data.frame(g1,g2,g3,g4)

predicted=c()

predictions=c()

Pred=c()

for(i in 1:4){

mod=glm(TennisDataAll.Result~.,data=PCdata[Groups[,i],c(1,2,3,21)]) ## note that we are grabbing the first and second columns of PCdata (PC1 and PC2) and the last column of PCdata (y)

predicted=predict(mod,newdata=PCdata[-Groups[,i],c(1,2,3,21)])

Pred=(fitted(mod)>.5)\*1

predictions=c(predictions,Pred)

}

sum(TennisDataAll$Result!=predictions)/length(predictions)

Getting the PCA Equation:

PCdata=data.frame(pca$x,TennisDataAll$Result, TennisDataAll$Hard\_or\_Soft, TennisDataAll$M\_or\_F)

glm(TennisDataAll.Result~.,data=PCdata[,c(1,2,3,21,22,23)])

Decision Trees:

Model 1: - Default CP

library(rpart)

library(rpart.plot)

TennisTree1 <- rpart(Who\_Won~. ,data = TennisDataAll)

rpart.plot(TennisTree1)

printcp(TennisTree1,digits=5)

plotcp(TennisTree1)

predictions1=predict(TennisTree1,type="class")

sum(TennisDataAll$Who\_Won!=predictions1)/484

g1 <- 1:121

g2<- 122:242

g3 <- 243:363

g4 <- 364:484

Groups <- data.frame(g1,g2,g3,g4)

predictions=c()

for(i in 1:4){

tree=rpart(Who\_Won~., data=TennisDataAll[-Groups[,i],] )

predictions\_per\_fold=predict(tree,type="class",newdata=TennisDataAll[Groups[,i],])

predictions=c(predictions,as.character(predictions\_per\_fold))

}

sum(TennisDataAll$Who\_Won!=predictions)/length(predictions)

Model 2: - CP of .015

TennisTree2 <- rpart(Who\_Won~. ,data = TennisDataAll, control = rpart.control(cp = .015))

rpart.plot(TennisTree2,digits=-3)

predictions2=predict(TennisTree2,type="class")

sum(TennisDataAll$Who\_Won!=predictions2)/484

predictions=c()

for(i in 1:4){

tree=rpart(Who\_Won~., data=TennisDataAll[-Groups[,i],] ,control = rpart.control(cp = .015))

predictions\_per\_fold=predict(tree,type="class",newdata=TennisDataAll[Groups[,i],])

predictions=c(predictions,as.character(predictions\_per\_fold))

}

sum(TennisDataAll$Who\_Won!=predictions)/length(predictions)

Model 3: - CP of .05

TennisTree3 <- rpart(Who\_Won~. ,data = TennisDataAll, control = rpart.control(cp = .05))

rpart.plot(TennisTree3,digits=-3)

predictions3=predict(TennisTree3,type="class")

sum(TennisDataAll$Who\_Won!=predictions3)/484

predictions=c()

for(i in 1:4){

tree=rpart(Who\_Won~., data=TennisDataAll[-Groups[,i],] ,control = rpart.control(cp = .05))

predictions\_per\_fold=predict(tree,type="class",newdata=TennisDataAll[Groups[,i],])

predictions=c(predictions,as.character(predictions\_per\_fold))

}

sum(TennisDataAll$Who\_Won!=predictions)/length(predictions)

Support Vector Machine (SVM):

Model 1: - Radial and Cost of 1

g1 <- 1:121

g2<- 122:242

g3 <- 243:363

g4 <- 364:484

Groups <- data.frame(g1,g2,g3,g4)

predictions=c()

for(i in 1:4){

model <- svm(Who\_Won~FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1,data=TennisDataAll[-Groups[,i],], kernel="radial",scale=TRUE,type="C-classification",cost=1)

group\_prediction <- predict(model, newdata = TennisDataAll[Groups[,i],])

predictions <- c(predictions,as.character(group\_prediction))

}

misclass <- sum(TennisDataAll$Who\_Won!=predictions)/nrow(TennisDataAll)

Model 2: - Radial and Cost of 100

predictions=c()

for(i in 1:4){

model = svm(Who\_Won~FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1,data=TennisDataAll[-Groups[,i],], kernel="radial",scale=TRUE,type="C-classification",cost=100)

group\_prediction = predict(model, newdata = TennisDataAll[Groups[,i],])

predictions=c(predictions,as.character(group\_prediction))

}

misclass=sum(TennisDataAll$Who\_Won!=predictions)/nrow(TennisDataAll)

Model 3: - Linear and Cost of 1

predictions=c()

for(i in 1:4){

model = svm(Who\_Won~FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1,data=TennisDataAll[-Groups[,i],], kernel="linear",scale=TRUE,type="C-classification",cost=1)

group\_prediction = predict(model, newdata = TennisDataAll[Groups[,i],])

predictions=c(predictions,as.character(group\_prediction))

}

misclass=sum(TennisDataAll$Who\_Won!=predictions)/nrow(TennisDataAll)

Model 4: - Linear and Cost of 100

predictions=c()

for(i in 1:4){

model = svm(Who\_Won~FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1,data=TennisDataAll[-Groups[,i],], kernel="linear",scale=TRUE,type="C-classification",cost=100)

group\_prediction = predict(model, newdata = TennisDataAll[Groups[,i],])

predictions=c(predictions,as.character(group\_prediction))

}

misclass=sum(TennisDataAll$Who\_Won!=predictions)/nrow(TennisDataAll)

Plots of the SVM (ACE and DBF):

svmTennis1<- svm(Who\_Won~ FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1 ,data=TennisDataAll, kernel="linear",scale=TRUE,type="C-classification",cost=1)

plot(svmTennis1,TennisDataAll,formula = ACE.1~ACE.2)

svmTennis1<- svm(Who\_Won~ FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1 ,data=TennisDataAll, kernel="linear",scale=TRUE,type="C-classification",cost=1)

plot(svmTennis1,TennisDataAll,formula = DBF.2 + DBF.1)

svmTennis1<- svm(Who\_Won~ FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1 ,data=TennisDataAll, kernel="linear",scale=TRUE,type="C-classification",cost=100)

plot(svmTennis1,TennisDataAll,formula = ACE.1~ACE.2)

svmTennis1<- svm(Who\_Won~ FSP.1+ FSP.2 + SSW.1 + SSW.2 + ACE.1 + ACE.2 + DBF.2 + DBF.1 ,data=TennisDataAll, kernel="linear",scale=TRUE,type="C-classification",cost=100)

plot(svmTennis1,TennisDataAll,formula = DBF.2 + DBF.1)

K Nearest Neighbors (KNN):

Standardizing:

for(i in 1:(ncol(TennisDataAll)-1)){

TennisDataAll[,i]=(TennisDataAll[,i]-mean(TennisDataAll[,i]))/sd(TennisDataAll[,i])

}

Model 1: - 5, 10, 15, 20… 50 Neighbors

g1 <- 1:121

g2<- 122:242

g3 <- 243:363

g4 <- 364:484

Groups <- data.frame(g1,g2,g3,g4)

misclass=c()

for (k in 5\*(1:50)){

Pred=c()

for(i in 1:4){

xs\_KNOWN\_CLASS=TennisDataAll[-Groups[,i],-22]

y\_KNOWN\_CLASS=TennisDataAll[-Groups[,i],22]

xs\_UNKNOWN\_CLASS=TennisDataAll[Groups[,i],-22]

PredTest=knn(train=xs\_KNOWN\_CLASS,test=xs\_UNKNOWN\_CLASS,cl=y\_KNOWN\_CLASS,k=k)

Pred <- c(Pred,as.character(PredTest))

}

misclass=c(misclass,sum(TennisDataAll$Who\_Won!= Pred)/484)

}

misclass

min(misclass)

Model 2: - 1 to 50 Neighbors

misclass=c()

for (k in 1\*(1:50)){

Pred=c()

for(i in 1:4){

xs\_KNOWN\_CLASS=TennisDataAll[-Groups[,i],-22]

y\_KNOWN\_CLASS=TennisDataAll[-Groups[,i],22]

xs\_UNKNOWN\_CLASS=TennisDataAll[Groups[,i],-22]

PredTest=knn(train=xs\_KNOWN\_CLASS,test=xs\_UNKNOWN\_CLASS,cl=y\_KNOWN\_CLASS,k=k)

Pred <- c(Pred,as.character(PredTest))

}

misclass=c(misclass,sum(TennisDataAll$Who\_Won!= Pred)/484)

}

Plot Code:

plot(5\*(1:50), misclass,pch=16)

plot(1\*(1:50), misclass,pch=16)

Clustering:

Standardazation:

for(k in 1:ncol(TennisDataCluster)){

TennisDataCluster[,k]=(TennisDataCluster[,k]-mean(TennisDataCluster[,k]))/sd(TennisDataCluster[,k])

}

Set Up:

Eucl.Distances= dist(TennisDataCluster)

Manh.Distances= dist(TennisDataCluster,method="manhattan")

AMClusters=hclust(Manh.Distances,method="average")

AEClusters=hclust(Eucl.Distances,method="average")

CMClusters=hclust(Manh.Distances,method="complete")

CEClusters=hclust(Eucl.Distances,method="complete")

Model 1: - Average Manhattan

GroupsAM=cutree(AMClusters,k=2)

Pred=rep("Home",486)

sum(TennisDataAll$Who\_Won[GroupsAM==1]=="Home") #241

sum(TennisDataAll$Who\_Won[GroupsAM==1]=="Away") #243

Pred[GroupsAM==1]="Away"

sum(TennisDataAll$Who\_Won[GroupsAM==2]=="Home") #2

sum(TennisDataAll$Who\_Won[GroupsAM==2]=="Away") #0

Pred[GroupsAM==2]="Home"

missclass=sum(TennisDataAll$Who\_Won!=Pred)/486

Model 2: - Complete Manhatan

GroupsCM=cutree(CMClusters,k=2)

sum(TennisDataAll$Who\_Won[GroupsCM==1]=="Home") #199

sum(TennisDataAll$Who\_Won[GroupsCM==1]=="Away") #200

Pred=rep("Home",486)

Pred[GroupsCM==1]="Away"

sum(TennisDataAll$Who\_Won[GroupsCM==2]=="Home") #44

sum(TennisDataAll$Who\_Won[GroupsCM==2]=="Away") #43

Pred[GroupsCM==2]="Home"

missclass=sum(TennisDataAll$Who\_Won!=Pred)/486

Model 3: - Average Euclidian

GroupsAE=cutree(AEClusters,k=2)

sum(TennisDataAll$Who\_Won[GroupsAE==1]=="Home") #241

sum(TennisDataAll$Who\_Won[GroupsAE==1]=="Away") #241

Pred=rep("Home",486)

Pred[GroupsAE==1]="Away"

sum(TennisDataAll$Who\_Won[GroupsAE==2]=="Home") #2

sum(TennisDataAll$Who\_Won[GroupsAE==2]=="Away") #2

Pred[GroupsAE==2]="Home"

missclass=sum(TennisDataAll$Who\_Won!=Pred)/486

Model 4: - Complete Euclidian

GroupsCE=cutree(CEClusters,k=2)

sum(TennisDataAll$Who\_Won[GroupsCE==1]=="Home") #188

sum(TennisDataAll$Who\_Won[GroupsCE==1]=="Away") #201

Pred=rep("Home",486)

Pred[GroupsCE==1]="Away"

sum(TennisDataAll$Who\_Won[GroupsCE==2]=="Home") #55

sum(TennisDataAll$Who\_Won[GroupsCE==2]=="Away") #42

Pred[GroupsCE==2]="Home"

missclass=sum(TennisDataAll$Who\_Won!=Pred)/486

Random Forests: - 1000 Trees

library(randomForest)

TennisDataAll$Who\_Won<- as.factor(TennisDataAll$Who\_Won)

RF=randomForest(Who\_Won~.,data=TennisDataAll, mtry = .5\*10, ntree = 1000)

predictions=predict(RF)

misclass <- sum(TennisDataAll$Who\_Won!=predictions)/484