STAT 4011 – Statistical Project (Part I)
Fraud Detection of Insurance Claims

Appendix & Reference

Group 10

Chin Yan Yi 1155078130
Law Lap Fei 1155079694
Li JianLiang 1155072287
Yuen Chun Wing 1155077960

X. Appendix

Recode the variables:

```
setwd("/Users/TyrusYuen/Google Drive/Sync/Year 4 Sem 1/STAT 4011/Project 
1")
rawdata<-read.csv("claims.csv",header=TRUE)

library(car)
library(fastDummies)
rawdata$Month<-recode(rawdata$Month,"'Jan'=1 ;'Feb'=2;
'Mar'=3;'Apr'=4;'May'=5;'Jun'=6;'Jul'=7;'Aug'=8;'Sep'=9;'Oct'=10;'Nov'=11;
'Dec'=12")
rawdata$MonthClaimed<-recode(rawdata$MonthClaimed,"'Jan'=1 ;'Feb'=2;
'Mar'=3;'Apr'=4;'May'=5;'Jun'=6;'Jul'=7;'Aug'=8;'Sep'=9;'Oct'=10;'Nov'=11;
'Dec'=12")
rawdata$DayOfWeek<-recode(rawdata$DayOfWeek,"'Monday'=2;'Tuesday'=3;'Wedne
sday'=4;'Thursday'=5; 'Friday'=6;'Saturday'=7;'Sunday'=1")
rawdata$DayOfWeekClaimed<-recode(rawdata$DayOfWeekClaimed,"'Monday'=2;'Tue
sday'=3;'Wednesday'=4;'Thursday'=5; 'Friday'=6;'Saturday'=7;'Sunday'=1")
rawdata$Sex<-recode(rawdata$Sex,"'Male'=1;'Female'=0")
rawdata$PastNumberOfClaims<-recode(rawdata$PastNumberOfClaims,"'none'=0")
rawdata$NumberOfSuppliments<-recode(rawdata$NumberOfSuppliments,"'none'=0"
)
rawdata$AgentType<-recode(rawdata$AgentType,"'External'=0;'Internal'=1")
rawdata$Fault<-recode(rawdata$Fault,"'Policy Holder'=0;'Third Party'=1")
rawdata$AccidentArea<-recode(rawdata$AccidentArea,"'Urban'=0;'Rural'=1")
rawdata$PoliceReportFiled<-recode(rawdata$PoliceReportFiled,"'Yes'=1;'No'=
0")
rawdata$WitnessPresent<-recode(rawdata$WitnessPresent,"'Yes'=1;'No'=0")

rawdata$Make<-recode(rawdata$Make,"'Lexus'=1;'Ferrari'=2;'Mecedes'=3;'Porc
he'=4;'Jaguar'=5;
'BMW'=6;'Nisson'=7;'Saturn'=8;'Mercury'=9;'Saab'=10;'Dodge'=11;'VW'=12;'Fo
rd'=13;'Accura'=14;'Chevrolet'=15;'Mazda'=16;'Honda'=17;'Toyota'=18;'Ponti
ac'=19")
rawdata$MaritalStatus<-recode(rawdata$MaritalStatus,"'Widow'=1;'Divorced'=
2;'Single'=3;'Married'=4")
rawdata$PolicyType<-recode(rawdata$PolicyType,"'Sport - 
Liability'=1;'Utility - Liability'=2;'Sport - All Perils'=3;'Utility - 
Collision'=4;'Utility - All Perils'=5;'Sport - Collision'=6;'Sedan - All 
Perils'=7;'Sedan - Liability'=8;'Sedan - Collision'=9")
rawdata$BasePolicy<-recode(rawdata$BasePolicy,"'All 
Perils'=1;'Liability'=2;'Collision'=3")
rawdata$VehicleCategory<-recode(rawdata$VehicleCategory,"'Utility'=1;'Spor
t'=2;'Sedan'=3")
rawdata$VehiclePrice<-recode(rawdata$VehiclePrice,"'20000 to 
29000'=2;'30000 to 39000'=3;'40000 to 59000'=4;'60000 to 69000'=5;'less 
than 20000'=1;'more than 69000'=6")
rawdata$VehiclePrice<-recode(rawdata$VehiclePrice,"'20000 to 
29000'=2;'30000 to 39000'=3;'40000 to 59000'=4;'60000 to 69000'=5;'less 
than 20000'=1;'more than 69000'=6")
rawdata$Days_Policy_Accident<-recode(rawdata$Days_Policy_Accident,"'none'=
1;'1 to 7'=2;'15 to 30'=3;'8 to 15'=4;'more than 30'=5")
rawdata$Days_Policy_Claim<-recode(rawdata$Days_Policy_Claim,"'none'=1;'1 
to 7'=2;'15 to 30'=3;'8 to 15'=4;'more than 30'=5")
rawdata$PastNumberOfClaims<-recode(rawdata$PastNumberOfClaims,"'0'=1;'1'=2
```

```
;'2 to 4'=3;'more than 4'=4;")
rawdata$AgeOfVehicle<-recode(rawdata$AgeOfVehicle,"'new'=1;'2 years'=2;'3
years'=3;'4 years'=4;'5 years'=5;'6 years'=6;'7 years'=7;'more than 7'=8")
rawdata$AgeOfPolicyHolder<-recode(rawdata$AgeOfPolicyHolder,"'16 to
17'=1;'18 to 20'=2;'21 to 25'=3;'26 to 30'=4;'31 to 35'=5;'36 to 40'=6;'41
to 50'=7;'51 to 65'=8; 'over 65'=9")
rawdata$NumberOfSuppliments<-recode(rawdata$NumberOfSuppliments,"'0'=1;'1
to 2'=2;'3 to 5'=3;'more than 5'=4;")
rawdata$AddressChange_Claim<-recode(rawdata$AddressChange_Claim,"'no
change'=1;'under 6 months'=2;'1 year'=3;'2 to 3 years'=4;'4 to 8
years'=5;")
rawdata$NumberOfCars<-recode(rawdata$NumberOfCars,"'1 vehicle'=1;'2
vehicles'=2;'3 to 4'=3;'5 to 8'=4;'more than 8'=5;")

#30 includes 'Under 6 months'
#15, 24, 25, 29, 30,31 included 'more than'/'Over'
#15, 21, 22, 23, 29, 31 are interval
#17 & 18 are ID number, not included in analysis


write.csv(rawdata,file="Count_nodatedata.csv")

#obtain the date by excel after recoding
```

<u>Correlation between the variables:</u>

```
library(GoodmanKruskal)
library(ggplot2)
library(caret)
library(mlbench)
gmk_count<-GoodmanKruskal::GKtauDataframe(count[,10:ncol(count)])
gmk_dummy<-GoodmanKruskal::GKtauDataframe(dummy[,9:ncol(dummy)])
count1 <- read.xlsx("3_regression selection.xlsx")
count2 <- read.xlsx("4_random forest selection.xlsx")
ggpairs(count2)
ggpairs(count1)
ggpairs(count)
```

<u>Principal Component Analysis:</u>

```
library(utils)
library(openxlsx)
claimdata<-read.xlsx("2_count_standardize.xlsx")
idpnt<-claimdata[,2:ncol(claimdata)]
(pca_count<-princomp(idpnt, cor = TRUE, scale. = TRUE) )
(vars <- (pca_count$sdev)^2)  #
???pca?????????????????(pca$sdev)????????????????????variance(?????????)

(props <- vars / sum(vars))

(cumulative.props <- cumsum(props))  # ?????????n???????????????

plot(cumulative.props)
# ??????plot()??????
plot(pca_count,          # ???pca
     type="line", # ????????????????????????
     main="Scree Plot for Count") # ?????????
```

```
# ???????????????????????????=1?????????
abline(h=1, col="blue") # Kaiser eigenvalue-greater-than-one rule
```

<u>Variable Selection:</u>

```
library(openxlsx)
count <- read.xlsx("2_count_standardize.xlsx")

model_count<-glm(formula = FraudFound_P ~ ., family = binomial(link =
"logit"), data = count)
model_count
summary(model_count)

#Results:

#Extra: Feature selection #Take super long time
# load the library
library(mlbench)
library(caret)

newcount<-read.xlsx('3_regression selection.xlsx')
idpnt<-newcount[,c(2:ncol(newcount))]
depnt<-newcount[,1]

# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(idpnt,depnt, sizes=c(1:15), rfeControl=control)
# summarize the results
print(results)
# list the chosen features
predictors(results)
# plot the results
plot(results, type=c("g", "o"))

count1 <- read.xlsx("3_regression selection.xlsx")
count2 <- read.xlsx("4_random forest selection.xlsx")
ggpairs(count2)
ggpairs(count1)
ggpairs(count)
```

<u>Logistic regression:</u>

```
cv.glm <-
  function (data, model=origin~., yname="origin", K=10) {
    datay=data[,yname]
    library(MASS)



    conmat<-matrix(c(0,0,0,0),nrow=2)
    result<-matrix(c(0,0,0,0),nrow=2)
```

```
    TN=NULL
    FP=NULL
    FN=NULL
    TP=NULL
    for (i in 1:K) { #i=1

glm.fit=glm(model,data=data[-draws[,i],],family=binomial)

      glm.y <- data[, yname]

glm.pred=predict(glm.fit,data[draws[,i],],type="response"
)
      glm.class=rep("0",nrow(data[draws[,i],]))
      glm.class[qda.pred>0.05]="1"

result<-(as.matrix(table(glm.y[draws[,i]],glm.class)))
      TN=c(TN,result[1,1])
      FP=c(FP,result[1,2])
      FN=c(FN,result[2,1])
      TP=c(TP,result[2,2])

conmat<-(as.matrix(table(glm.y[draws[,i]],glm.class)))+co
nmat

    }
    print(conmat)
    print("TN")
    print(c(summary(TN),"SD=",sd(TN)))
    print("FP")
    print(c(summary(FP),"SD=",sd(FP)))
    print("FN")
    print(c(summary(FN),"SD=",sd(FN)))
    print("TP")
    print(c(summary(TP),"SD=",sd(TP)))

    #Output
    list( K = K
         )
  }
```

<u>FLDA:</u>

```
cv.lda <-
  function (data, model=origin~., yname="origin", K=10) {
    datay=data[,yname]
    library(MASS)
```

```
    conmat<-matrix(c(0,0,0,0),nrow=2)
    result<-matrix(c(0,0,0,0),nrow=2)
    TN=NULL
    FP=NULL
    FN=NULL
    TP=NULL
    for (i in 1:K) { #i=1



      lda.fit=lda(model, data=data[-draws[,i],])

      lda.y <- data[, yname]

      lda.pred=predict(lda.fit, data[draws[,i],])

      lda.class=lda.pred$class


result<-(as.matrix(table(lda.y[draws[,i]],lda.class)))
      TN=c(TN,result[1,1])
      FP=c(FP,result[1,2])
      FN=c(FN,result[2,1])
      TP=c(TP,result[2,2])




conmat<-(as.matrix(table(lda.y[draws[,i]],lda.class)))+co
nmat



    }
    print(conmat)
    print("TN")
    print(c(summary(TN),"SD=",sd(TN)))
    print("FP")
    print(c(summary(FP),"SD=",sd(FP)))
    print("FN")
    print(c(summary(FN),"SD=",sd(FN)))
    print("TP")
    print(c(summary(TP),"SD=",sd(TP)))

    #Output
    list( K = K
         )
  }
```

QDA:

```
cv.qda <-
  function (data, model=origin~., yname="origin", K=10) {
    datay=data[,yname]
    library(MASS)







    conmat<-matrix(c(0,0,0,0),nrow=2)
    result<-matrix(c(0,0,0,0),nrow=2)
    TN=NULL
    FP=NULL
    FN=NULL
    TP=NULL
    for (i in 1:K) { #i=1



      qda.fit=qda(model, data=data[-draws[,i],])

      qda.y <- data[, yname]

      qda.pred=predict(qda.fit, data[draws[,i],])

      qda.class=qda.pred$class


result<-(as.matrix(table(qda.y[draws[,i]],qda.class)))
      TN=c(TN,result[1,1])
      FP=c(FP,result[1,2])
      FN=c(FN,result[2,1])
      TP=c(TP,result[2,2])





conmat<-(as.matrix(table(qda.y[draws[,i]],qda.class)))+co
nmat




    }
    print(conmat)
    print("TN")
    print(c(summary(TN),"SD=",sd(TN)))
    print("FP")
    print(c(summary(FP),"SD=",sd(FP)))
    print("FN")
    print(c(summary(FN),"SD=",sd(FN)))
    print("TP")
```

```
                        print(c(summary(TP),"SD=",sd(TP)))

                        #Output
                        list( K = K
                             )
                      }
```

<u>Nearest Neighbour Algorithm:</u>

```
library(class)
library(caret)# loading the data
library(openxlsx)
set.seed(10)
fdt<-read.xlsx('2_count_standardize.xlsx')
trainntest<-read.csv('draws.csv')
trainntest<-as.matrix(trainntest[2:nrow(trainntest),1:ncol(trainntest)])

# creating matrices for Xs and Y
responseY <- as.matrix(fdt[,1])
X <- as.matrix(fdt[,c(2:ncol(fdt))])



# data partition for train/test sets. # data partition for 1 train/test
sets. Test k=1:30
trainIndex <- createDataPartition(responseY, times=1, p = 0.8, list = F)

#Result

# fitting models for 30 different k-values (one for test and one for train
set for each K)
train.error = rep(0,30)
test.error = rep(0,30)
for(k in 1:30){
  model.knn.train <- knn(train=X[trainIndex,], test=X[trainIndex,],
cl=responseY[trainIndex], k=k, prob=F)
  train.error[k] <-
sum(model.knn.train!=responseY[trainIndex])/length(responseY[trainIndex])
  model.knn.test <- knn(train=X[trainIndex,], test=X[-trainIndex,],
cl=responseY[trainIndex], k=k, prob=F)
  test.error[k] <-
sum(model.knn.test!=responseY[-trainIndex])/length(responseY[-trainIndex])
}

# PLOTTING:
plot(1:30, train.error, col='red', type = 'b', ylim = c(0,0.65))
points(1:30, test.error, col='blue', type = 'b')
title("Error for k=1:30, 1 trial", sub = "Red= Train; Blue= Test",
      cex.main = 2,   font.main= 4, col.main= "blue",
      cex.sub = 0.75, font.sub = 3, col.sub = "red")
#Result: 1

# data partition for 10 train/test sets. Test k=1:30
train.error = c()
test.error = c()
```

```
for(k in 1:30){
  test.error.tmp = c()
  train.error.tmp = c()
  for(i in 1:10){
    pred <- knn(train = X[-trainntest[,i],],test = X[trainntest[,i],], cl
= responseY[-trainntest[,i],], k=k)
    test.error.tmp = c(test.error.tmp,mean(responseY[trainntest[,i],] !=
pred))
    pred <- knn(train = X[-trainntest[,i],],test = X[-trainntest[,i],], cl
= responseY[-trainntest[,i],], k=k)
    train.error.tmp = c(train.error.tmp,mean(responseY[trainntest[,i],] !=
pred))
  }
  test.error = rbind(test.error,test.error.tmp)
  train.error = rbind(train.error,train.error.tmp)
}
plot(1:30, rowMeans(train.error), col='blue', type='b', ylim = c(0,0.65))
points(1:30, rowMeans(test.error), col='red', type='b')
title("Error for k=1:30, 10 trial", sub = "Red= Train; Blue= Test",
      cex.main = 2,   font.main= 4, col.main= "blue",
      cex.sub = 0.75, font.sub = 3, col.sub = "red")

# data partition for 10 train/test sets. Show the confusion matrix where
k=3



train.error = c()
test.error = c()
conmat<-matrix(c(0,0,0,0),nrow=2)
for(i in 1:10){
  pred <- knn(train = X[-trainntest[,i],],test = X[trainntest[,i],], cl =
responseY[-trainntest[,i],], k=5)
  test.error.tmp = c(test.error.tmp,mean(responseY[trainntest[,i],] !=
pred))
  conmat<-as.matrix(table(pred, responseY[trainntest[,i]]))+conmat
  pred <- knn(train = X[-trainntest[,i],],test = X[-trainntest[,i],], cl =
responseY[-trainntest[,i],], k=5)
  train.error.tmp = c(train.error.tmp,mean(responseY[-trainntest[[i]],] !=
pred))
}
conmat

# k=1:30 result

train.error = c()
test.error = c()
test.error.tmp=c()
conmat<-matrix(c(0,0,0,0),nrow=2)
for (k in 1:10){
  conmat<-matrix(c(0,0,0,0),nrow=2)
  tp<-vector(mode='numeric')
  tn<-vector(mode='numeric')
  fp<-vector(mode='numeric')
  fn<-vector(mode='numeric')
```

```
  for(i in 1:10){
    pred <- knn(train = X[-trainntest[,i],],test = X[trainntest[,i],], cl
= responseY[-trainntest[,i],], k=k)
    test.error.tmp = c(test.error.tmp,mean(responseY[trainntest[,i],] !=
pred))

conmat<-t(as.matrix(table(responseY[trainntest[,i]],pred,dnn=c('predict',p
aste('actual',k)))))+conmat
    sensitivity<-conmat[2,2]/(conmat[2,2]+conmat[2,1])

accuracy<-(conmat[2,2]+conmat[1,1])/(conmat[2,1]+conmat[2,2]+conmat[1,1]+c
onmat[1,2])
    precision<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
    specificity<-conmat[1,1]/(conmat[1,1]+conmat[1,2])
    ppv<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
    npv<-conmat[1,1]/(conmat[1,1]+conmat[2,1])

performance1<-t(as.matrix(c(accuracy,precision,sensitivity,specificity,ppv
,npv)))
    tp<-c(tp,conmat[2,2])
    tn<-c(tn,conmat[1,1])
    fp<-c(fp,conmat[2,1])
    fn<-c(fn,conmat[1,2])
  }
  print(conmat)

colnames(performance1)<-c('Accuracy','Precision','Sensitivity','Specificit
y','Positive Predictive Value','Negative Predictive Value')
  print(performance1)
}

# k=1:30 result

train.error = c()
test.error = c()
test.error.tmp=c()
conmat<-matrix(c(0,0,0,0),nrow=2) #initialize the matrix needed
conmat1<-matrix(c(0,0,0,0),nrow=2)
  tp<-vector(mode='numeric') #ininitialize the vector storing the elements
of confusion matrix
  tn<-vector(mode='numeric')
  fp<-vector(mode='numeric')
  fn<-vector(mode='numeric')
  for(i in 1:10){
    pred <- knn(train = X[-trainntest[,i],],test = X[trainntest[,i],], cl
= responseY[-trainntest[,i],], k=1) #Run knn
    test.error.tmp = c(test.error.tmp,mean(responseY[trainntest[,i],] !=
pred))

conmat<-t(as.matrix(table(responseY[trainntest[,i]],pred,dnn=c('predict',p
aste('actual',1))))) #Confusion matrix storing the ith trial reuslt

conmat1<-t(as.matrix(table(responseY[trainntest[,i]],pred,dnn=c('predict',
paste('actual',1)))))+conmat1 #Confusion matrix storing 1-10 trials trsult
    print(conmat) #print the ith trial
```

```
    sensitivity<-conmat[2,2]/(conmat[2,2]+conmat[2,1])

accuracy<-(conmat[2,2]+conmat[1,1])/(conmat[2,1]+conmat[2,2]+conmat[1,1]+c
onmat[1,2])
    precision<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
    specificity<-conmat[1,1]/(conmat[1,1]+conmat[1,2])
    ppv<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
    npv<-conmat[1,1]/(conmat[1,1]+conmat[2,1])

performance1<-t(as.matrix(c(accuracy,precision,sensitivity,specificity,ppv
,npv)))

colnames(performance1)<-c('Accuracy','Precision','Sensitivity','Specificit
y','Positive Predictive Value','Negative Predictive Value')
     print(performance1)
    tp<-c(tp,conmat[2,2])

    tn<-c(tn,conmat[1,1])

    fp<-c(fp,conmat[2,1])

    fn<-c(fn,conmat[1,2])

  }
  print(conmat1)
  print(c(summary(tp),sd(tp)))
  print(c(summary(tn),sd(tn)))
  print(c(summary(fp),sd(fp)))
  print(c(summary(fn),sd(fn)))
  sensitivity1<-conmat[2,2]/(conmat[2,2]+conmat[2,1])

accuracy1<-(conmat[2,2]+conmat[1,1])/(conmat[2,1]+conmat[2,2]+conmat[1,1]+
conmat[1,2])
  precision1<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
  specificity1<-conmat[1,1]/(conmat[1,1]+conmat[1,2])
  ppv1<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
  npv1<-conmat[1,1]/(conmat[1,1]+conmat[2,1])

performance2<-t(as.matrix(c(accuracy1,precision1,sensitivity1,specificity1
,ppv1,npv1)))

colnames(performance2)<-c('Accuracy','Precision','Sensitivity','Specificit
y','Positive Predictive Value','Negative Predictive Value')
  print(performance2)
```

Classification Tree

```
#Without Pruning tree

library(rpart)
library(rpart.plot)
library(openxlsx)

X2_count_standardize<-read.xlsx("Desktop/STAT4011/Final/2_count_standardiz
e.xlsx")
```

```
X<-X2_count_standardize

responseY<-as.matrix(X[,1])
predictorX<-as.matrix(X[,c(2:ncol(X))])
trainntest<-read.xlsx("Desktop/STAT4011/Final/draws.xlsx")

  for (K in 1:10){
    conmat<-matrix(c(0,0,0,0),nrow=2)
    TN<-NULL
    FP<-NULL
    FN<-NULL
    TP<-NULL
  for(i in 1:10){
    set.seed(1002)
    r_i<-rpart(FraudFound_P ~., data=X[-trainntest[,i],],
method="class",control=rpart.control(cp=0.0001))
    pred <- predict(r_i, X[trainntest[,i],], type="class")

conmat<-(as.matrix(table(responseY[trainntest[,i]],pred,dnn=c('actual',pas
te('predict',k)))))+conmat

    TN<-c(TN,conmat[1,1])
    FP<-c(FP,conmat[1,2])
    FN<-c(FN,conmat[2,1])
    TP<-c(TP,conmat[2,2])

    sensitivity<-conmat[2,2]/(conmat[2,2]+conmat[2,1])

accuracy<-(conmat[2,2]+conmat[1,1])/(conmat[2,1]+conmat[2,2]+conmat[1,1]+c
onmat[1,2])
    precision<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
    specificity<-conmat[1,1]/(conmat[1,1]+conmat[1,2])

performance1<-t(as.matrix(c(sensitivity,accuracy,precision,specificity)))
  }

colnames(performance1)<-c('sensitivity','accuracy','precision','specificit
y')
    print(performance1)
    print("TN")
    print(c(summary(TN),"SD=",sd(TN)))
    print("FP")
    print(c(summary(FP),"SD=",sd(FP)))
    print("FN")
    print(c(summary(FN),"SD=",sd(FN)))
    print("TP")
    print(c(summary(TP),"SD=",sd(TP)))
  }
conmat
plotcp(r_i)
plot_i<-rpart.plot(r_i, type=2, digits=3, fallen.leaves = TRUE)

#With prune tree

library(rpart)
```

```
library(rpart.plot)
library(openxlsx)

X2_count_standardize<-read.xlsx("Desktop/STAT4011/Final/2_count_standardiz
e.xlsx")
X<-X2_count_standardize

responseY<-as.matrix(X[,1])
predictorX<-as.matrix(X[,c(2:ncol(X))])
trainntest<-read.xlsx("Desktop/STAT4011/Final/draws.xlsx")

for (K in 1:10){
  conmat<-matrix(c(0,0,0,0),nrow=2)
  TN<-NULL
  FP<-NULL
  FN<-NULL
  TP<-NULL
  for(i in 1:10){
    set.seed(1002)
    r_i<-rpart(FraudFound_P ~., data=X[-trainntest[,i],],
method="class",control=rpart.control(cp=0.0001))
    printcp(r_i)
    # Select the tree size that minimizes misclassification rate (i.e.
prediction error).
    # Prediction error rate in training data (resubstitution error rate)=
Root node error * rel error
    # Prediction error rate in cross-validation = Root node error * xerror
    # Select the cp value (construct a simpler tree) that minimizes the
xerror.

    bestcp <- r_i$cptable[which.min(r_i$cptable[,"xerror"]),"CP"]
    r_i.prune <- prune(r_i, cp = bestcp)

    pred <- predict(r_i.prune, X[trainntest[,i],], type="class")

conmat<-(as.matrix(table(responseY[trainntest[,i]],pred,dnn=c('actual',pas
te('predict',k)))))+conmat

    TN<-c(TN,conmat[1,1])
    FP<-c(FP,conmat[1,2])
    FN<-c(FN,conmat[2,1])
    TP<-c(TP,conmat[2,2])

    sensitivity<-conmat[2,2]/(conmat[2,2]+conmat[2,1])

accuracy<-(conmat[2,2]+conmat[1,1])/(conmat[2,1]+conmat[2,2]+conmat[1,1]+c
onmat[1,2])
    precision<-conmat[2,2]/(conmat[2,2]+conmat[1,2])
    specificity<-conmat[1,1]/(conmat[1,1]+conmat[1,2])

performance1<-t(as.matrix(c(sensitivity,accuracy,precision,specificity)))
  }

colnames(performance1)<-c('sensitivity','accuracy','precision','specificit
y')
```

```
  print(performance1)
  print("TN")
  print(c(summary(TN),"SD=",sd(TN)))
  print("FP")
  print(c(summary(FP),"SD=",sd(FP)))
  print("FN")
  print(c(summary(FN),"SD=",sd(FN)))
  print("TP")
  print(c(summary(TP),"SD=",sd(TP)))
}
conmat
plotcp(r_i.prune)
plot_i<-rpart.plot(r_i.prune, type=2, digits=3, fallen.leaves = TRUE)
```

Artificial Neural Network:

```
setwd("~/Google Drive/Sync/Year 4 Sem 1/STAT 4011/Project 1")
library(tidyverse)
library(neuralnet)
library(GGally)
library(class)
library(caret)# loading the data
library(openxlsx)
library(nnet)
library(kernlab)
set.seed(10)
fdt<-read.xlsx('3_regression selection.xlsx')
#ggpairs(fdt,title='Scatterplot Matrix of the features of the dataset')
trainntest<-read.csv('draws.csv')
trainntest<-as.matrix(trainntest[2:nrow(trainntest),1:ncol(trainntest)])

conmat<-matrix(c(0,0,0,0),nrow=2)
  for(i in 1:10){
rbf <- rbfdot(sigma=0.1)
irisSVM <-
ksvm(FraudFound_P~.,data=fdt[-trainntest[,i],],type="C-bsvc",kernel=rbf,C=
10,prob.model=TRUE)
fitted(irisSVM)
pdt_svm1<-predict(irisSVM, fdt[trainntest[,i],2:ncol(fdt)],
type="response")
conmat<-conmat+table(pdt_svm1,fdt[trainntest[,i],1])
}
table(pdt_svm1,fdt[trainntest[,1],1])

#nnet
train<-fdt[trainntest[,1],]
fdt_NN1 <- nnet(fdt[,2:ncol(fdt)],fdt[,1], size=10)
summary(fdt_NN1)
pred<-round(fdt_NN1$fit)
table(pred,fdt[,1])
pdt_NN1<-predict(fdt_NN1, fdt[trainntest[,1],2:ncol(fdt)], type="raw")
table(pdt_NN1,fdt[trainntest[,1],1])

#neuralnet
```

```
fdt_NeurN1<- neuralnet(formula = FraudFound_P     ~
Date+WeekNumber+Dateclaimed+Month+PolicyType+VehicleCategory+BasePolicy,fd
t[-trainntest[,1],], linear.output = FALSE,err.fct='ce',likelihood=TRUE)

fdt_NeurN1_trainerror<-fdt_NeurN1$result.matrix[1,1]
paste('CE Error: ',round(fdt_NeurN1_trainerror,3)) #CE Error

fdt_NeurN1_AIC<-fdt_NeurN1$result.matrix[4,1]
paste('AIC: ',round(fdt_NeurN1_AIC,3))

fdt_NeurN1_BIC<-fdt_NeurN1$result.matrix[5,1]
paste('BIC: ',round(fdt_NeurN1_BIC,3))

main=glm(FraudFound_P~
Date+WeekNumber+Dateclaimed+Deductible+Year+Month+WeekOfMonth+DayOfWeek+Ac
cidentArea+Sex+Fault+PolicyType+VehicleCategory+VehiclePrice+Days_Policy_A
ccident+PoliceReportFiled+AddressChange_Claim+BasePolicy,fdt[trainntest[,1
],],family=binomial())
prediction(fdt_NeurN1,list.glm = list(main=main))

compute(fdt_NeurN1,fdt[trainntest[,1],2:ncol(fdt)])
```

XI. References

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

Chawla, N. (n.d.). DATA MINING FOR IMBALANCED DATASETS: AN OVERVIEW. Retrieved October 21, 2018, from https://www3.nd.edu/~dial/publications/chawla2005data.pdf

Elliot, M., Fairweather, I., Olsen, W., & Pampaka, M. (2016). Oversampling. A Dictionary of Social Research Methods, A Dictionary of Social Research Methods.

Fraud Investigator Salaries in the United States. (2018, October 17). Retrieved October 21, 2018, from https://www.indeed.com/salaries/Fraud-Investigator-Salaries

James, G., Witten, D., Hastie, T., Tibshirani, R. (2017). An Introduction to Statistical Learning with Applications in R: Springer.

Lemaître, G. K., Nogueira, F., & Aridas, C. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research, 18, 1-5.

Patel, N. P., Sarraf, E. H., & Tsai, M. (2018). The Curse of Dimensionality. Anesthesiology, 129(3), 614-615.

Rahman, M., & Davis, D. N. (2013). Addressing the Class Imbalance Problem in Medical Datasets. International Journal of Machine Learning and Computing, 3(2), 224-228. doi:10.18411/a-2017-023

Seoung-Hun Park, & Young-Guk Ha. (2014). Large Imbalance Data Classification Based on MapReduce for Traffic Accident Prediction. Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2014 Eighth International Conference on, 45-49.

Tomar, A., & Nagpal, A. (2016). Comparing Accuracy of K-Nearest-Neighbor and Support-Vector-Machines for Age Estimation. International Journal of Engineering Trends and Technology, 38(6), 326-329. doi:10.14445/22315381/ijett-v38p260

Yam, S. C. (2018). RMSC4002_Combined_Lecture_Notes[Class handout]. Hong Kong: the Chinese University of Hong Kong, RMSC 4002.