STAT 574 HOMEWORK 2

Derrick Edwards

Problem 1. Use the data in the file "hospital_data.csv" to fit a random forest regression. Display variable importance and compute prediction accuracy within 10%, 15%, and 20% of the true values.

R

```
> print(importance(randfor,type=2))
                     IncNodePurity
gender                   254905796
age                     2293999649
BMI                     1892621958
ASA                      257809531
surgery_duration_min   24937769989


> print("Accuracy Scores")
[1] "Accuracy Scores"
> print(mean(accuracy10))
[1] 0.5263819
> print(mean(accuracy15))
[1] 0.7173367
> print(mean(accuracy20))
[1] 0.8266332
```

R Code

```
## Problem 1 ##################################################################
#############################################################################

library(randomForest)

# use hospital data to fit random forest regression
 # display variable importance
 # compute prediction accuracy within 10%, 15%, 20%

hospital <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/hospital_data.csv")

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(109283)
sample <- sample(c(TRUE, FALSE), nrow(hospital),
          replace = TRUE, prob = c(0.8, 0.2))
train <- hospital[sample,]
test <- hospital[!sample,]

summary(hospital)
```

```
# build random forest regression model
randfor <- randomForest(surgery_cost ~ gender + age + BMI + ASA +
                    surgery_duration_min, data=train, ntree=150,
                   mtry=5, maxnodes=30)

# display variable importance #######
#####################################
print(importance(randfor,type=2))

# compute prediction accuracy ######
#####################################
# computing prediction accuracy for testing data
p_surg_cost <- predict(randfor, newdata = test)

# accuracy 10,15, 20 store true false values - compute means for accuracy scores

# accuracy within 10%
accuracy10 <- ifelse(abs(test$surgery_cost - p_surg_cost)
              < 0.10*test$surgery_cost,1,0)

# accuracy within 15%
accuracy15 <- ifelse(abs(test$surgery_cost - p_surg_cost)
              < 0.15*test$surgery_cost,1,0)
# accuracy within 20%
accuracy20 <- ifelse(abs(test$surgery_cost - p_surg_cost)
              < 0.20*test$surgery_cost,1,0)

# print means of accuracy scores
print("Accuracy Scores")
print(mean(accuracy10))
print(mean(accuracy15))
print(mean(accuracy20))
```

Python

```
var_name loss_reduction
4 surgery_duration_min 0.638316
2 BMI 0.175907
1 age 0.136569
3 ASA 0.026618
0 gender 0.022591

Accuracy Scores
0.5183727034120735 – 10%
```

0.6889763779527559 – 15%
0.8110236220472441 – 20%

Python Code

```
# Problem 1 - random forest regression  - display variable importance and
accuracy scores within 10%, 15%, 20%

import pandas
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

# read in data
hospital=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\hospital_data.csv')
coding={'M': 1, 'F': 0}
hospital['gender']=hospital['gender'].map(coding)

X=hospital.iloc[:,1:6].values
y=hospital.iloc[:,6].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=348644)

#FITTING RANDOM FOREST REGRESSION TREE
rf_reg=RandomForestRegressor(n_estimators=100, random_state=323445,
max_depth=50, max_features=4)
rf_reg.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['gender','age','BMI','ASA','surgery_duration_min'],
columns=['var_name'])
loss_reduction=pandas.DataFrame(rf_reg.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)


#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=rf_reg.predict(X_test)
```

```
ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print(accuracy10)

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print(accuracy15)

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print(accuracy20)
```

SAS

Loss Reduction Variable Importance

Variable	Number of Rules	MSE	OOB MSE	Absolute Error	OOB Absolute Error
surgery_duration_min	58569	11567664	5624273	1313.861728	409.822479
gender	807	147890	-40140	32.807187	7.906966
ASA	5221	428920	-214740	89.422811	-9.460088
age	22887	2065206	-1230355	432.589289	-97.107737
BMI	39848	2603059	-2518896	592.624166	-238.879936

The SAS System

accuracy10	accuracy15	accuracy20
0.501971	0.668857	0.781866

SAS Code

```
proc import out=hospital
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\hospital_data.csv"
dbms=csv replace;
run;

*proc print data=hospital;
*run;
```

```
/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING*/
proc surveyselect data=hospital rate=0.8 seed=502305
out=hosptialNew outall method=srs;
run;


/* random forest regression model */
proc hpforest data=hosptialNew seed=109283
maxtrees=60 vars_to_try=4 trainfraction=0.7
maxdepth=50;
target surgery_cost/level=interval;
input gender/level=nominal;
input age BMI ASA surgery_duration_min/level=interval;
partition rolevar=selected(train='1');
save file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin';
run;


/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set hosptialNew;
if(selected='0');
run;


proc hp4score data=test;
id surgery_cost;
score file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin'
out=predicted;
run;

/*DETERMINING 10%, 15%, AND 20% ACCURACY*/
data accuracy;
set predicted;
if(abs(surgery_cost-P_surgery_cost)
<0.10*surgery_cost)
then ind10=1; else ind10=0;
if(abs(surgery_cost-P_surgery_cost)
<0.15*surgery_cost)
then ind15=1; else ind15=0;
if(abs(surgery_cost-P_surgery_cost)
<0.20*surgery_cost)
then ind20=1; else ind20=0;
run;

proc sql;
 select sum(ind10)/count(*) as accuracy10,
sum(ind15)/count(*) as accuracy15,
 sum(ind20)/count(*) as accuracy20
 from accuracy;
 quit;
```

Problem 2. Use the data in the file "card_transdata.csv" to build a random forest binary classifier. Display variable importance and compute prediction accuracy.

R

```
                                MeanDecreaseGini
distance_from_home                    36.9290625
distance_from_last_transaction        11.4447027
ratio_to_median_purchase_price       133.0867253
repeat_retailer                        0.8906286
used_chip                             14.7836872
used_pin_number                       19.7203744
online_order                          45.0600826

> print(mean(accuracy))
[1] 0.9954442
```

R Code

```
## Problem 2 ##########################################################################
##############################################################################
# card data
# build random forest binary classifier
# display variable importance and prediction accuracy

card_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data
Mining/HW1STAT574S23/DATA SETS/card_transdata.csv")

# split data 80% train and 20% test
set.seed(229120)
sample <- sample(c(TRUE,FALSE), nrow(card_data),
           replace=TRUE, prob=c(0.8,0.2))
train <- card_data[sample,]
test <- card_data[!sample,]

summary(card_data)
# build random forest binary classifier
library(randomForest)
rf_class <- randomForest(as.factor(fraud) ~ distance_from_home +
                   distance_from_last_transaction +
                   ratio_to_median_purchase_price + repeat_retailer +
                   used_chip + used_pin_number + online_order,
                  data = train, ntree=150, mtry=4, maxnodes=30)

# display feature importance
```

```
print(importance(rf_class, type=2))

# compute prediction accuracy for testing data
predclass <- predict(rf_class, newdata = test)
test <- cbind(test, predclass)

accuracy <- c()
for (i in 1:nrow(test)) {
  accuracy[i] <- ifelse(test$fraud[i]==test$predclass[i],1,0)
}

print(mean(accuracy))
```

Python

```
Accuracy: 0.9975
```

Python Code

```
# Problem 2 - random forest binary classifier - variable importance and
prediction accuracy
import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

card_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\card_transdata.csv')
X=card_data.iloc[:,0:7].values
y=card_data.iloc[:,7]

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=786756)

#FITTING RANDOM FOREST BINARY CLASSIFIER
rf_class=RandomForestClassifier(n_estimators=150, criterion='entropy',
random_state=778554, max_depth=50, max_features=4)
rf_class.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier
```

```
var_names=pandas.DataFrame(['distance_from_home','distance_from_last_transaction'
, 'ratio_to_median_purchase_price',
'repeat_retailer', 'used_chip','used_pin_number','online_order'],
columns=['var_names'])
loss_reduction=pandas.DataFrame(rf_class.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)


# print accuracy score
from sklearn.metrics import accuracy_score
# store predicted values from testing set
y_pred=rf_class.predict(X_test)


accuracy=accuracy_score(y_test, y_pred)

print("Accuracy: ", accuracy)
```

SAS

Loss Reduction Variable Importance

Variable	Number of Rules	Gini	OOB Gini	Margin	OOB Margin
ratio_to_median_purchase_price	420	0.077661	0.06277	0.155322	0.140032
online_order	141	0.034680	0.03373	0.069359	0.068706
distance_from_home	292	0.021442	0.01044	0.042884	0.032071
used_pin_number	60	0.007936	0.00771	0.015873	0.015452
used_chip	68	0.007464	0.00744	0.014929	0.014609
distance_from_last_transaction	319	0.012214	-0.00004	0.024428	0.012392
repeat_retailer	12	0.000421	-0.00006	0.000842	0.000532

accuracy
0.9875

SAS Code

```
proc import out=card_data
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\card_transdata.csv"
dbms=csv replace;
run;
```

```
/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=card_data rate=0.8 seed=1029384
out=card_dataNew outall method=srs;
run;

/*BUILDING RANDOM FOREST BINARY CLASSIFIER*/
proc hpforest data=card_dataNew seed=115607
maxtrees=60 vars_to_try=4 trainfraction=0.7
maxdepth=50;
target fraud/level=binary;
input repeat_retailer used_chip used_pin_number online_order/level=binary;
input distance_from_home distance_from_last_transaction
      ratio_to_median_purchase_price/level=interval;
partition rolevar=selected(train='1');
save file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin';
run;


/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set card_dataNew;
if(selected='0');
run;

proc hp4score data=test;
id fraud;
score file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin'
out=predicted;
run;

/*COMPUTING PREDICTION ACCURACY FOR TESTING DATA*/
data predicted;
set predicted;
match=(fraud=lowcase(I_fraud));
run;

proc sql;
select sum(match)/count(*) as accuracy
from predicted;
quit;
```

Problem 3. Use the data in the file "concussions_data.csv" to construct a random forest multinomial classifier. Display variable importance and compute prediction accuracy.

R

```
> print(importance(rf_multi_class, type=2))
              MeanDecreaseGini
```

```
age                   13.88135
nyearsplaying         11.55411
position              73.09288
prevconc             143.48521

> print(mean(accuracy))
[1] 0.9137931
```

R Code

```
## Problem 3 ###################################################################
#################################################################################

# random forest multinomial classifier
 # variable importance and prediction accuracy

concuss <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/concussions_data.csv")

# split data 80% train 20% test
set.seed(223494)
sample <- sample(c(TRUE,FALSE), nrow(concuss), replace=TRUE,
                 prob=c(0.8,0.2))
train <- concuss[sample,]
test <- concuss[!sample,]

summary(concuss)

# building random forest multinomial classifier
library(randomForest)
rf_multi_class <- randomForest(as.factor(concussion) ~ age + nyearsplaying +
                                 position + prevconc, data = train, ntree=150,
                               mtry=4, maxnodes=30)


# variable importance
print(importance(rf_multi_class, type=2))

# prediction accuracy for testing data
predclass <- predict(rf_multi_class, newdata = test)
test <- cbind(test, predclass)

accuracy <- c()
for (i in 1:nrow(test)) {
  accuracy[i] <- ifelse(test$concussion[i]==test$predclass[i],1,0)
}

print(mean(accuracy))
```

Python

```
var_names loss_reduction
3 prevconc 0.484487
2 position 0.399035
0 age 0.062954
1 nyearsplaying 0.053523

Accuracy: 0.8666666666666667
```

Python Code

```
# Problem 3
    # construct random forest multinomial classifier - display variable
importance and accuracy
import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

concussion_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\concussions_data.csv')
#for col in concussion_data:
#    print(concussion_data[col].unique())

code_position={'Offensive Lineman': 1, 'Cornerback': 2, 'Running Back': 3,'Wide
Receiver': 4,
'Quarterback': 5}
code_concussion={'mild': 1, 'moderate': 2, 'severe': 3}

concussion_data['position']=concussion_data['position'].map(code_position)
concussion_data['concussion']=concussion_data['concussion'].map(code_concussion)


X=concussion_data.iloc[:,0:4]
y=concussion_data.iloc[:,4]

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=599555)

#FITTING RANDOM FOREST FOR MULTINOMIAL CLASSIFIER
rf_class=RandomForestClassifier(n_estimators=150, random_state=663474,
max_depth=50, max_features=4)
rf_class.fit(X_train, y_train)
```

```
#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['age','nyearsplaying','position','prevconc'],
columns=['var_names'])
loss_reduction=pandas.DataFrame(rf_class.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)

# print accuracy score
from sklearn.metrics import accuracy_score
# store predicted values from testing set
y_pred=rf_class.predict(X_test)

accuracy=accuracy_score(y_test, y_pred)

print("Accuracy: ", accuracy)
```

Loss Reduction Variable Importance

Variable	Number of Rules	Gini	OOB Gini	Margin	OOB Margin
prevconc	453	0.282217	0.28239	0.321634	0.31620
position	441	0.251870	0.23963	0.476527	0.46600
age	2004	0.032747	-0.02126	0.065493	0.01682
nyearsplaying	1774	0.028614	-0.03550	0.057229	-0.00017

accuracy
0.865385

SAS Code

```
proc import out=concuss
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\concussions_data.csv"
dbms=csv replace;
run;

proc print data=concuss;
run;
```

```
/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=concuss rate=0.8 seed=550040
out=concussNew outall method=srs;
run;

/*BUILDING RANDOM FOREST MULTINOMIAL CLASSIFIER*/
proc hpforest data=concussNew seed=454545
maxtrees=150 vars_to_try=4 trainfraction=0.7
maxdepth=10;
target concussion/level=nominal;
input position/level=nominal;
input age nyearsplaying prevconc/level=interval;
partition rolevar=selected(train='1');
save file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin';
run;



/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set concussNew;
if(selected='0');
run;

proc hp4score data=test;
id concussion;
score file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin'
out=predicted;
run;

/*COMPUTING PREDICTION ACCURACY FOR TESTING DATA*/
data predicted;
set predicted;
match=(concussion=lowcase(I_concussion));
run;

proc sql;
select sum(match)/count(*) as accuracy
from predicted;
quit;
```

Problem 4. Use the data in the file "hospital_data.csv" to fit a gradient boosted regression. Display variable importance and compute prediction accuracy within 10%, 15%, and 20% of the true values.

R

Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
surgery_duration_min	0.66992506	0.33217842	0.20283793

Feature	Gain	Cover	Frequency
<chr>	<dbl>	<dbl>	<dbl>
MedID	0.11578523	0.21664418	0.28149606
BMI	0.09505910	0.19183406	0.22591317
age	0.07909414	0.19406390	0.19739720
ASA	0.02493410	0.03957746	0.04967738
gender	0.01520236	0.02570197	0.04267826

```
> print(mean(accuracy10))
[1] 0.5448799
> print(mean(accuracy15))
[1] 0.7332491
> print(mean(accuracy20))
[1] 0.8268015
```

R Code

```
## Problem 4 ##############################################################
##########################################################################

# fit gradient boosted regression - display variable importance
  # and compute prediction accuracy within 10%, 15%, 20%
library(xgboost)

hospital <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data
Mining/HW1STAT574S23/DATA SETS/hospital_data.csv")

# split data 80% train and 20% test
set.seed(1029374)
sample <- sample(c(TRUE, FALSE), nrow(hospital), replace=TRUE,
                 prob=c(0.8,0.2))
train <- hospital[sample,]
test <- hospital[!sample,]

# numerical value is dependent variable
train.x<- data.matrix(train[-7])
train.y<- data.matrix(train[7])
test.x<- data.matrix(test[-7])
test.y<- data.matrix(test[7])

# fit extreme gradient boosted regression tree
xgb_reg <- xgboost(data = train.x, label = train.y, max.depth=6, eta=0.01,
                   subsample=0.8, colsample_bytree=0.5, nrounds=1000,
                   objective="reg:linear")
# NOTES xgboost function R
  # eta=learning rate
  #colsample_bytree = defines what percentage of features/columns will be used
    # for building each tree
```

```
# display feature importance
print(xgb.importance(colnames(train.x), model = xgb_reg))

# compute prediction accuracy for testing data
pred.y <- predict(xgb_reg, test.x)

# accuracy scores
# 10%
accuracy10 <- ifelse(abs(test.y-pred.y) < 0.10*test.y,1,0)
# 15%
accuracy15 <- ifelse(abs(test.y-pred.y) < 0.15*test.y,1,0)
# 20%
accuracy20 <- ifelse(abs(test.y-pred.y) < 0.20*test.y,1,0)


# print accuracy scores
print(mean(accuracy10))
print(mean(accuracy15))
print(mean(accuracy20))
```

Python

```
var_name loss_reduction
4 surgery_duration_min 0.717264
2 BMI 0.130330
1 age 0.113203
0 gender 0.020439
3 ASA 0.018765

0.541994750656168 – 10%
0.7073490813648294 – 15%
0.8110236220472441 – 20%
```

Python Code

```
# Problem 4
    # fit gradient boosted regression - display variable importance and accuracy
scores 10,15, 20%

import pandas
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split

# read in data
```

```
hospital=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\hospital_data.csv')
coding={'M': 1, 'F': 0}
hospital['gender']=hospital['gender'].map(coding)

X=hospital.iloc[:,1:6].values
y=hospital.iloc[:,6].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=348644)

#FITTING GRADIENT BOOSTED REGRESSION TREE
gbreg_params = {'n_estimators': 1000, 'max_depth': 6, 'learning_rate': 0.01,
'loss': 'squared_error'}
gb_reg=GradientBoostingRegressor(**gbreg_params)
gb_reg.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE

var_names=pandas.DataFrame(['gender','age','BMI','ASA','surgery_duration_min'],
columns=['var_name'])
loss_reduction=pandas.DataFrame(gb_reg.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)


#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=gb_reg.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print(accuracy10)
```

```
#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print(accuracy15)

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print(accuracy20)
```

SAS

The SAS System

accuracy10	accuracy15	accuracy20
0.503937	0.681102	0.788714

SAS Code

```
proc import out=sasuser.hospital
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\hospital_data.csv"
dbms=csv replace;
run;


/*Gradient boosted regression model is built
in Enterprise Miner*/



libname hw2q1 "\\vdi-
fileshare01\UEMprofiles\017365554\Desktop\XGBoostReg\Workspaces\EMWS1\emsave"
;

data accuracy;
set hw2q1.em_save_test;
ind10=(abs(R_surgery_cost)<0.10*surgery_cost);
ind15=(abs(R_surgery_cost)<0.15*surgery_cost);
ind20=(abs(R_surgery_cost)<0.20*surgery_cost);
run;

proc sql;
select sum(ind10)/count(*) as accuracy10,
sum(ind15)/count(*) as accuracy15,
sum(ind20)/count(*) as accuracy20
from accuracy;
quit;
```

1)

SAS

Loss Reduction Variable Importance

Variable	Number of Rules	MSE	OOB MSE	Absolute Error	OOB Absolute Error
surgery_duration_min	58569	11567664	5624273	1313.861728	409.822479
gender	807	147890	-40140	32.807187	7.906966
ASA	5221	428920	-214740	89.422811	-9.460088
age	22887	2065206	-1230355	432.589289	-97.107737
BMI	39848	2603059	-2518896	592.624166	-238.879936

The SAS System

accuracy10	accuracy15	accuracy20
0.501971	0.668857	0.781866

SAS Code

```
proc import out=hospital
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\hospital_data.csv"
dbms=csv replace;
run;

*proc print data=hospital;
*run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING*/
proc surveyselect data=hospital rate=0.8 seed=502305
out=hosptialNew outall method=srs;
run;


/* random forest regression model */
proc hpforest data=hosptialNew seed=109283
maxtrees=60 vars_to_try=4 trainfraction=0.7
maxdepth=50;
target surgery_cost/level=interval;
input gender/level=nominal;
input age BMI ASA surgery_duration_min/level=interval;
partition rolevar=selected(train='1');
save file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin';
run;


/*COMPUTING PREDICTED VALUES FOR TESTING DATA*/
data test;
set hosptialNew;
if(selected='0');
run;
```

```
proc hp4score data=test;
id surgery_cost;
score file='\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\random_forest.bin'
out=predicted;
run;

/*DETERMINING 10%, 15%, AND 20% ACCURACY*/
data accuracy;
set predicted;
if(abs(surgery_cost-P_surgery_cost)
<0.10*surgery_cost)
then ind10=1; else ind10=0;
if(abs(surgery_cost-P_surgery_cost)
<0.15*surgery_cost)
then ind15=1; else ind15=0;
if(abs(surgery_cost-P_surgery_cost)
<0.20*surgery_cost)
then ind20=1; else ind20=0;
run;

proc sql;
 select sum(ind10)/count(*) as accuracy10,
sum(ind15)/count(*) as accuracy15,
 sum(ind20)/count(*) as accuracy20
 from accuracy;
 quit;
```

Problem 5. Use the data in the file "card_transdata.csv" to build a gradient boosted binary classifier. Display variable importance and compute prediction accuracy.

R

Feature <chr>	Gain <dbl>	Cover <dbl>	Frequency <dbl>
ratio_to_median_purchase_price	0.490308699	0.30038158	0.314341211
distance_from_home	0.181529759	0.25000378	0.287795403
online_order	0.154217388	0.09943558	0.052444157
distance_from_last_transaction	0.083715743	0.22049302	0.261573325
used_pin_number	0.057100078	0.06047156	0.022661055
used_chip	0.029985248	0.05382343	0.053415345
repeat_retailer	0.003143086	0.01539105	0.007769505

```
> print(mean(match))
[1] 0.9949239
```

R Code

```
## Problem 5 ##########################################################
##############################################################################

# gradient boosted binary classifier
 # display variable importance and compute prediction accuracy
library(xgboost)

card_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/card_transdata.csv")

# split data 80% train 20% test
set.seed(573920)
sample <- sample(c(TRUE, FALSE), nrow(card_data), replace=TRUE,
                 prob=c(0.8,0.2))
train <- card_data[sample,]
test <- card_data[!sample,]

train.x<- data.matrix(train[-8])
train.y<- data.matrix(train[8])
test.x<- data.matrix(test[-8])
test.y<- data.matrix(test[8])

# fit gradient boosted binary classifier
xgb_class <- xgboost(data=train.x, label = train.y, max.depth=6, eta=0.1,
                     subsample=0.8, colsample_bytree=0.5, nrounds = 1000,
                     objective="binary:logistic")

# display feature importance
print(xgb.importance(colnames(train.x), model = xgb_class))

# prediction accuracy for testing data
pred.prob <- predict(xgb_class, test.x)

len <- length(pred.prob)
pred.fraud <- c()
match <- c()
for (i in 1:len) {
  pred.fraud[i] <- ifelse(pred.prob[i]>=0.5,1,0)
  match[i] <- ifelse(test.y[i]==pred.fraud[i],1,0)
}

print(mean(match))
```

Python

```
var_names loss_reduction
2 ratio_to_median_purchase_price 0.434809
```

```
6 online_order 0.248841
0 distance_from_home 0.122693
5 used_pin_number 0.107861
4 used_chip 0.062697
1 distance_from_last_transaction 0.023098
3 repeat_retailer 0.000000

Accuracy: 0.9975
```

Python Code

```
# Problem 5
    # gradient boosted binary classifier - variable importance and prediction
accuracy

import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier

card_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\card_transdata.csv')
X=card_data.iloc[:,0:7].values
y=card_data.iloc[:,7]

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=786756)

#FITTING GRADIENT BOOSTED BINARY CLASSIFIER
gbclass_params = {'n_estimators': 1000, 'max_depth': 6, 'learning_rate': 0.1}
gb_class=GradientBoostingClassifier(**gbclass_params)
gb_class.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['distance_from_home','distance_from_last_transaction'
, 'ratio_to_median_purchase_price',
'repeat_retailer', 'used_chip','used_pin_number','online_order'],
columns=['var_names'])
loss_reduction=pandas.DataFrame(gb_class.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)
```

```
# print accuracy score
from sklearn.metrics import accuracy_score
# store predicted values from testing set
y_pred=gb_class.predict(X_test)


accuracy=accuracy_score(y_test, y_pred)

print("Accuracy: ", accuracy)
```

SAS

accuracy
1

SAS Code

```
proc import out=sasuser.card_data
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\card_transdata.csv"
dbms=csv replace;
run;



/*Gradient boosted regression model is built
in Enterprise Miner*/

libname hw2q1 "\\vdi-
fileshare01\UEMprofiles\017365554\Desktop\XGBoostBin\Workspaces\EMWS1\emsave"
;

data hw2q1.em_save_test;
set hw2q1.em_save_test;
match=(EM_CLASSIFICATION=EM_CLASSTARGET);
run;

proc sql;
select sum(match)/count(*) as accuracy
from hw2q1.em_save_test;
quit;
```

Problem 6. Use the data in the file “concussions_data.csv” to construct a gradient boosted multinomial classifier. Display variable importance and compute prediction accuracy.

R

Feature	Gain	Cover	Frequency
<chr>	<dbl>	<dbl>	<dbl>
prevconc	0.4122571	0.09663928	0.05918021
position	0.3417179	0.17313137	0.13382246
age	0.1364528	0.38297640	0.42899588
nyearsplaying	0.1095722	0.34725295	0.37800146

```
> print(mean(match))
[1] 0.8651685
```

R Code

```
## Problem 6 ######################################################################
################################################################################

# gradient boosted multinomial classifier
  # variable importance and prediction accuracy
library(xgboost)
concuss <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data
Mining/HW1STAT574S23/DATA SETS/concussions_data.csv")

# split data 80% train 20% test
set.seed(749385)
sample <- sample(c(TRUE,FALSE), nrow(concuss), replace=TRUE,
          prob=c(0.8,0.2))
train <- concuss[sample,]
test <- concuss[!sample,]

train.x<- data.matrix(train[-5])
train.y<- data.matrix(train[5])
train.y<- train.y-1 #must range between 0 and 4 for prediction
test.x<- data.matrix(test[-5])
test.y<- data.matrix(test[5])
test.y<- test.y-1

# fit graident boosted multinomial classifier
xgb_multi_class <- xgboost(data=train.x, label = train.y, max.depth=6,
                   eta=0.1, subsample=0.8, colsample_bytree=0.5,
                   nrounds=1000, num_class=5,
                   objective="multi:softprob")

#DISPLAYING FEATURE IMPORTANCE
print(xgb.importance(colnames(train.x), model=xgb_multi_class))


# compute prediction accuracy test data
pred.prob <- predict(xgb_multi_class, test.x, reshape=TRUE)
```

```
pred.prob <- as.data.frame(pred.prob)
colnames(pred.prob) <- 0:2

pred.class <- apply(pred.prob, 1, function(x)
  colnames(pred.prob)[which.max(x)])

match <- c()
for(i in 1:length(test.y)){
  match[i] <- ifelse(pred.class[i]==as.character(test.y[i]),1,0)
}

print(mean(match))
```

Python

```
var_names loss_reduction
3 prevconc 0.532275
2 position 0.386110
0 age 0.046393
1 nyearsplaying 0.035222

Accuracy: 0.8380952380952381
```

Python Code

```
# Problem 6
    # gradient boosted multinomial classifier - variable importance and accuracy

import pandas
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier

concussion_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\concussions_data.csv')
#for col in concussion_data:
#    print(concussion_data[col].unique())

code_position={'Offensive Lineman': 1, 'Cornerback': 2, 'Running Back': 3,'Wide
Receiver': 4,
'Quarterback': 5}
code_concussion={'mild': 1, 'moderate': 2, 'severe': 3}

concussion_data['position']=concussion_data['position'].map(code_position)
concussion_data['concussion']=concussion_data['concussion'].map(code_concussion)

X=concussion_data.iloc[:,0:4]
```

```
y=concussion_data.iloc[:,4]

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=566033)

#FITTING GRADIENT BOOSTED MULTINOMIAL CLASSIFIER
gbmclass_params = {'n_estimators': 1000, 'max_depth': 6, 'learning_rate': 0.1}
gb_mclass=GradientBoostingClassifier(**gbmclass_params)
gb_mclass.fit(X_train, y_train)

#DISPLAYING VARIABLE IMPORTANCE
from sklearn.ensemble import ExtraTreesClassifier

var_names=pandas.DataFrame(['age','nyearsplaying','position','prevconc'],
columns=['var_names'])
loss_reduction=pandas.DataFrame(gb_mclass.feature_importances_,
columns=['loss_reduction'])
var_importance=pandas.concat([var_names, loss_reduction], axis=1)
var_importance=var_importance.sort_values("loss_reduction", axis=0,
ascending=False)
print(var_importance)

# print accuracy score
from sklearn.metrics import accuracy_score
# store predicted values from testing set
y_pred=gb_mclass.predict(X_test)

accuracy=accuracy_score(y_test, y_pred)

print("Accuracy: ", accuracy)
```

SAS

accuracy
0.897196

SAS Code

```
proc import out=sasuser.concuss
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\concussions_data.csv"
dbms=csv replace;
run;



/*Gradient boosted regression model is built
```

```
in Enterprise Miner*/

libname hw2q1 "\\vdi-
fileshare01\UEMprofiles\017365554\Desktop\XGBoostMulti\Workspaces\EMWS1\emsav
e";

data hw2q1.em_save_test;
set hw2q1.em_save_test;
match=(EM_CLASSIFICATION=EM_CLASSTARGET);
run;

proc sql;
select sum(match)/count(*) as accuracy
from hw2q1.em_save_test;
quit;
```