Cory Suzuki

STAT 574

2/24/25

Homework 2

Problem 1

SAS Code and Outputs

```
/* STAT 574 HW2 Problem 1 */

proc import out=hospital
datafile="C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data.
csv"
dbms=csv replace;
run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING*/
proc surveyselect data=hospital rate=0.8 seed=233364
out=hospital outall method=srs;
run;

/*BUILDING RANDOM FOREST REGRESSION*/
proc hpforest data=hospital seed=520530
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='C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/random_forest.bin'
;
run;

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

proc hp4score data=test;
id surgery_cost;
score
file='C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/random_forest.bin'
out=predicted;
run;

proc print;
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;
```

The SAS System

accuracy10	accuracy15	accuracy20
0.509855	0.683311	0.805519

Python Code and Outputs

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split

hospital_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data.c
sv")
gender_code = {'M':1, 'F':0}
hospital_data['gender'] = hospital_data['gender'].map(gender_code)
X = hospital_data.iloc[:, 0:6].values
y = hospital_data.iloc[:, 6].values

#Splitting the 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=364470)
```

```python
#Fitting random forest regression tree.
rf_reg=RandomForestRegressor(n_estimators=100, random_state=444510,
max_depth=50, max_features=4)
rf_reg.fit(X_train, y_train)

#Displaying variable importance.

var_names=pd.DataFrame(['gender', 'age', 'BMI', 'ASA', 'surgery_duration_min',
      'surgery_cost'], columns=['var_name'])
loss_reduction=pd.DataFrame(rf_reg.feature_importances_,
columns=['loss_reduction'])
var_importance=pd.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)
```

```
                 var_name  loss_reduction
5            surgery_cost        0.607783
3                     ASA        0.129775
0                  gender        0.117167
2                     BMI        0.103555
4   surgery_duration_min        0.024810
1                     age        0.016910
0.5275590551181102
0.6771653543307087
0.7939632545931758
```

R Code and Outputs

```r
library(readr)
library(randomForest)

hospital_data =
read.csv(file="C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data
.csv",
header=T, sep=",")

# Splitting data into 80% training and 20% testing sets.

set.seed(364323)
sample = sample(c(T,F), nrow(hospital_data),
replace=T, prob=c(0.8, 0.2))
train = hospital_data[sample,]
test = hospital_data[!sample,]

# Building random forest regression.

rf_reg_hosp = randomForest(surgery_cost~age+BMI+ASA
+surgery_duration_min, data=train, ntree=150, mtry=5,
maxnodes=30)

# Displaying feature importance.

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

```r
# Computing prediction accuracy for testing set.

P_surgery_cost = predict(rf_reg_hosp, newdata=test)

# Accuracy within 10%

accuracy10 = ifelse(abs(test$surgery_cost-P_surgery_cost)<0.10*test$surgery_cost,
1, 0)
print(accuracy10 <- mean(accuracy10))

# Accuracy within 15%

accuracy15 = ifelse(abs(test$surgery_cost-P_surgery_cost)<0.15*test$surgery_cost,
1, 0)
print(accuracy15 <- mean(accuracy15))

# Accuracy within 20%

accuracy20 = ifelse(abs(test$surgery_cost-P_surgery_cost)<0.20*test$surgery_cost,
1, 0)
print(accuracy20 <- mean(accuracy20))
```

```
                     IncNodePurity
age                     2308653769
BMI                     1845764432
ASA                      296242122
surgery_duration_min   28352723958
[1] 0.5442875
[1] 0.7175866
[1] 0.8267009
```

Problem 2

SAS Code and Outputs

```sas
/* STAT 574 HW2 Problem 2 */
```

```
proc import out=card_data
datafile="C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata
.csv"
dbms=csv replace;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=card_data rate=0.8 seed=328323
out=card_data outall method=srs;
run;

/*BUILDING RANDOM FOREST BINARY CLASSIFIER*/
proc hpforest data=card_data seed=115113
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=nominal;
input distance_from_home distance_from_last_transaction
ratio_to_median_purchase_price/level=interval;
partition rolevar=selected(train='1');
save
file='C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/random_forest.bin'
;
run;

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

proc hp4score data=test;
id fraud;
score
file='C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/random_forest.bin'
out=predicted;
run;

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

proc sql;
select mean(match) as accuracy
from predicted;
quit;
```

The SAS System

accuracy
0.995

Python Code and Outputs

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

card_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.
csv")
X = card_data.iloc[:, 0:7].values
y = card_data.iloc[:, 7].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=698498)

# Fitting random forest binary classifier.

rf_card_binary = RandomForestClassifier(n_estimators=150,
                                        criterion='entropy',
                                        random_state=233122,
                                        max_depth=50,
                                        max_features=7)

rf_card_binary.fit(X_train, y_train)

# Displaying variable importance.

var_names=pd.DataFrame(['distance_from_home',
    'distance_from_last_transaction', 'ratio_to_median_purchase_price',
    'repeat_retailer', 'used_chip', 'used_pin_number', 'online_order'],
columns=['var_name'])
```

```python
loss_reduction=pd.DataFrame(rf_card_binary.feature_importances_,
columns=['loss_reduction'])
var_importance=pd.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_card_binary.predict(X_test)
y_test=pd.DataFrame(y_test,columns=['fraud'])
y_pred=pd.DataFrame(y_pred,columns=['predicted'])
df=pd.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['fraud'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

accuracy=sum(match)/len(match)

print(accuracy)
```

```
                         var_name  loss_reduction
2  ratio_to_median_purchase_price        0.384837
0              distance_from_home        0.201095
6                    online_order        0.178875
5                  used_pin_number        0.098834
1  distance_from_last_transaction        0.088171
4                       used_chip        0.044444
3                 repeat_retailer        0.003744
0.9925
```

R Code and Outputs

```r
library(readr)
library(randomForest)
```

```r
card_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.csv
",
header=T, sep=",")

# Splitting the data into 80% training and 20% testing sets.

set.seed(474123)
sample = sample(c(T,F), nrow(card_data),
replace=T, prob=c(0.8, 0.2))
train = card_data[sample,]
test = card_data[!sample,]

# Building random forest binary classifier.

rf_bin_cls =
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)

# Displaying feature importance.

print(importance(rf_bin_cls, type=2))

# Computing prediction accuracy for testing data.

predclass = predict(rf_bin_cls, newdata=test)
test = cbind(test, predclass)

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

```
                                 MeanDecreaseGini
distance_from_home                     38.1579160
distance_from_last_transaction         12.7543747
ratio_to_median_purchase_price        120.9278202
repeat_retailer                         0.6828153
used_chip                              11.9989478
used_pin_number                        19.6896855
online_order                           46.5793895
[1] 0.9975962
```

Problem 3

SAS Code and Outputs

```
/* STAT 574 HW2 Problem 3 */

proc import out=concussion_data
datafile="C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_da
ta.csv"
dbms=csv replace;
run;

/*SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS*/
proc surveyselect data=concussion_data rate=0.8 seed=224113
out=concussion_data outall method=srs;
run;

/*BUILDING RANDOM FOREST MULTINOMIAL CLASSIFIER*/
proc hpforest data=concussion_data seed=177013
maxtrees=150 vars_to_try=3 trainfraction=0.9
maxdepth=50;
target concussion/level=ordinal;
input position prevconc/level=nominal;
input age nyearsplaying/level=interval;
partition rolevar=selected(train='1');
save
file='C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/random_forest.bin'
;
run;

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

proc hp4score data=test;
id concussion;
score
file='C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/random_forest.bin'
```

```
out=predicted;
run;

proc print;
run;

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

proc sql;
select mean(match) as accuracy
from predicted;
quit;
```

The SAS System

accuracy
0.875

Python Code

```
# STAT 574 HW2 Problem 3

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

concussion_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_dat
a.csv")
position_code = {'Offensive Lineman':0, 'Cornerback':1, 'Running Back':2,
'Quarterback':3, 'Wide Receiver':4}
concussion_code = {'mild':0, 'moderate':1, 'severe':2}
concussion_data['position'] = concussion_data['position'].map(position_code)
concussion_data['concussion'] =
concussion_data['concussion'].map(concussion_code)

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

```python
# Split 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=576485)

# Fitting random forest multinomial classifier.

rf_multi_cls = RandomForestClassifier(n_estimators=150,
        random_state=690233, max_depth=50, max_features=4)
rf_multi_cls.fit(X_train, y_train)

# Displaying variable importance.

var_names = pd.DataFrame(['age', 'nyearsplaying', 'position', 'prevconc'],
                         columns=['var_name'])
loss_reduction = pd.DataFrame(rf_multi_cls.feature_importances_,
columns=['loss_reduction'])
var_importance = pd.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_multi_cls.predict(X_test)
y_test=pd.DataFrame(y_test,columns=['concussion'])
y_pred=pd.DataFrame(y_pred,columns=['predicted'])
df=pd.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['concussion'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

accuracy=sum(match)/len(match)

print(accuracy)
```

```
           var_name  loss_reduction
3          prevconc        0.496968
2          position        0.377248
0               age        0.080299
1    nyearsplaying         0.045484
0.9333333333333333
```

R Code and Outputs

```
# STAT 574 HW2 Problem 3

library(readr)
library(randomForest)

concussion_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_data.c
sv",
header=T, sep=",")

# Splitting data into 80% training and 20% testing sets.

set.seed(333528)
sample = sample(c(T,F), nrow(concussion_data), replace=T,
prob=c(0.8, 0.2))
train = concussion_data[sample,]
test = concussion_data[!sample,]

# Building a multinomial random forest classifier.

rf_multi_class = randomForest(as.factor(concussion)~age+nyearsplaying
+position+prevconc, data=train, ntree=150, mtry=4, maxnodes=30)

# Displaying feature importance.

print(importance(rf_multi_class, type=2))

# Computing prediction accuracy from 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(accuracy <- mean(accuracy))
```

```
               MeanDecreaseGini
age                   13.770311
nyearsplaying          9.528452
position              82.559038
prevconc             128.986791
[1] 0.9152542
```

Problem 4

SAS Code and Outputs

```
data accuracy;
 set tmp1.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);

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

The SAS System

accuracy10	accuracy15	accuracy20
0.503937	0.681102	0.788714

Python Code and Outputs

```
# STAT 574 HW2 Problem 4

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

hospital_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data.c
sv")
gender_code = {'M':1, 'F':0}
hospital_data['gender'] = hospital_data['gender'].map(gender_code)
X = hospital_data.iloc[:, 0:6].values
y = hospital_data.iloc[:, 6].values

# Splitting the 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=471475)

# Fitting gradient boosting regression model.

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 = pd.DataFrame(['gender', 'age', 'BMI', 'ASA', 'surgery_duration_min',
      'surgery_cost'], columns=['var_name'])
```

```
loss_reduction=pd.DataFrame(gb_reg.feature_importances_,
columns=['loss_reduction'])
var_importance=pd.concat([var_names, loss_reduction], axis=1)
print(var_importance.sort_values("loss_reduction", axis=0, ascending=False))

# 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)
```

```
                var_name  loss_reduction
5           surgery_cost        0.685867
0                 gender        0.095421
2                    BMI        0.093783
3                    ASA        0.091865
1                    age        0.017077
4   surgery_duration_min        0.015986
0.520997375328084
0.6929133858267716
0.8044619422572179
```

R Code and Outputs

```
library(xgboost)

hospital_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/hospital_data.csv"
,
header=T, sep=",")

# Splitting the data into 80% training and 20% testing sets.

set.seed(698498)
sample = sample(c(T,F), nrow(hospital_data), replace=T, prob=c(0.8, 0.2))
train = hospital_data[sample,]
test = hospital_data[!sample,]

train_x = data.matrix(train[-6])
train_y = data.matrix(train[6])
test_x = data.matrix(test[-6])
test_y = data.matrix(test[6])

# Fitting 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,
```

```r
nrounds=1000, objective="reg:linear")

# Displaying Feature Importance

print(xgb.importance(colnames(train_x), model=xgb_reg))

# Computing prediction accuracy for testing data

pred_y = predict(xgb_reg, test_x)

# Accuracy within 10%

accuracy10 = ifelse(abs(test_y-pred_y)<0.10*test_y, 1, 0)
print(mean(accuracy10))

# Accuracy within 15%

accuracy15 = ifelse(abs(test_y-pred_y)<0.15*test_y, 1, 0)
print(mean(accuracy15))

# Accuracy within 20%

accuracy20 = ifelse(abs(test_y-pred_y)<0.20*test_y, 1, 0)
print(mean(accuracy20))
```

```
         Feature       Gain      Cover  Frequency
          <char>      <num>      <num>      <num>
1:  surgery_cost 0.69330644 0.37115021 0.24711126
2:         MedID 0.10237074 0.22312246 0.28155607
3:           BMI 0.09800160 0.20055253 0.20793441
4:           age 0.08302830 0.16033843 0.18075272
5:           ASA 0.01236230 0.02420521 0.04104765
6:        gender 0.01093062 0.02063115 0.04159789
[1] 0.4207077
[1] 0.5910878
[1] 0.7247706
```

Problem 5

SAS Code and Output

```
data tmp1.em_save_test;
set tmp1.em_save_test;
match=(EM_CLASSIFICATION=EM_CLASSTARGET);
run;

proc sql;
select sum(match)/count(*) as accuracy
from tmp1.em_save_test;
quit;
```

The SAS System

accuracy
1

Python Code and Outputs

```
# STAT 574 HW2 Problem 5

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

card_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.
csv")
X = card_data.iloc[:, 0:7].values
y = card_data.iloc[:, 7].values

# Splitting the 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=876424)

# Fitting a Binary Gradient Boosting Classifier.
```

```python
gbbinary_params = {'n_estimators': 1000, 'max_depth':7,
                   'learning_rate':0.1}
gbbinary_cls = GradientBoostingClassifier(**gbbinary_params)
gbbinary_cls.fit(X_train, y_train)

# Displaying variable importance.

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

# Computing prediction accuracy on testing data.

y_pred=gbbinary_cls.predict(X_test)
y_test=pd.DataFrame(y_test,columns=['fraud'])
y_pred=pd.DataFrame(y_pred,columns=['predicted'])
df=pd.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['fraud'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

accuracy=sum(match)/len(match)

print(accuracy)
```

```
                         var_name  loss_reduction
2  ratio_to_median_purchase_price        0.407621
6                    online_order        0.271699
5                 used_pin_number        0.118814
0              distance_from_home        0.112464
4                       used_chip        0.064915
1  distance_from_last_transaction        0.024487
3                 repeat_retailer        0.000000
0.9975
```

R Code and Outputs

```
library(xgboost)

card_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/card_transdata.csv
",
header=T, sep=",")

# Splitting data into 80% training and 20% testing sets.

set.seed(364663)
sample = sample(c(T,F), nrow(card_data), replace=T, prob=c(0.8, 0.2))
train = card_data[sample,]
test = card_data[!sample,]

train_x = data.matrix(train[-7])
train_y = data.matrix(train[7])
test_x = data.matrix(test[-7])
test_y = data.matrix(test[7])

# Fitting gradient boosted binary classifier.

xgb_bin = xgboost(data=train_x, label=train_y,
max.depth=8, eta=0.1, subsample=0.8, colsample_bytree=0.5,
nrounds=1000, objective="binary:logistic")

# Displaying feature importance.

print(xgb.importance(colnames(train_x), model=xgb_bin))
```

```
# Computing prediction accuracy for testing data.

pred_prob = predict(xgb_bin, test_x)

len = length(pred_prob)
pred_card = c()
match = c()
for (i in 1:len){
    pred_card[i] = ifelse(pred_prob[i]>=0.5, 1, 0)
    match[i] = ifelse(test_y[i]==pred_card[i], 1, 0)
}

print(prop <- sum(match)/len)
```

```
                              Feature        Gain       Cover   Frequency
                               <char>       <num>       <num>       <num>
 1:                distance_from_home 0.327141108 0.32581616 0.338495818
 2: ratio_to_median_purchase_price 0.318889609 0.30819644 0.311:
      distance_from_home 0.327141108 0.32581616 0.338495818
 2: ratio_to_median_purchase_price 0.318889609 0.30819644 0.318495818
 2: ratio_to_median_purchase_price 0.318889609 0.30819644 0.312: ratio_t
 o_median_purchase_price 0.318889609 0.30819644 0.318479015
 3: distance_from_last_transaction 0.288119872 0.27970687 0.293384958
 4:                             fraud 0.032298782 0.02670133 0.005661687
 5:                         used_chip 0.015654686 0.02366199 0.022865909
 6:                   used_pin_number 0.012901622 0.02062447 0.013: distanc
 e_from_last_transaction 0.288119872 0.27970687 0.293384958
 4:                             fraud 0.032298782 0.02670133 0.005661687
 5:                         used_chip 0.015654686 0.02366199 0.022865909
 6:                   used_pin_number 0.012901622 0.02062447 0.013384958
 4:                             fraud 0.032298782 0.02670133 0.005661687
 5:                         used_chip 0.015654686 0.02366199 0.022865909
 6:                   used_pin_number 0.012901622 0.02062447 0.015661687
 5:                         used_chip 0.015654686 0.02366199 0.022865909
 6:                   used_pin_number 0.012901622 0.02062447 0.015:
              used_chip 0.015654686 0.02366199 0.022865909
 6:                   used_pin_number 0.012901622 0.02062447 0.012865909
```

```
               used_chip 0.015654686 0.02366199 0.022865909
6:                   used_pin_number 0.012901622 0.02062447 0.012865909
6:                   used_pin_number 0.012901622 0.02062447 0.016:
         used_pin_number 0.012901622 0.02062447 0.013295832
3295832
7816781
[1] 0.6214834
```

Problem 6

SAS Code and outputs

```
data tmp1.em_save_test;
set tmp1.em_save_test;
match=(EM_CLASSIFICATION=EM_CLASSTARGET);
run;

proc sql;
select sum(match)/count(*) as accuracy
from tmp1.em_save_test;
quit;
```

The SAS System

accuracy
1

Python Code and Outputs

```
# STAT 574 HW2 Problem 6

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
```

```
concussion_data =
pd.read_csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_dat
a.csv")
position_code = {'Offensive Lineman':0, 'Cornerback':1, 'Running Back':2,
'Quarterback':3, 'Wide Receiver':4}
concussion_code = {'mild':0, 'moderate':1, 'severe':2}
concussion_data['position'] = concussion_data['position'].map(position_code)
concussion_data['concussion'] =
concussion_data['concussion'].map(concussion_code)

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

# Splitting the 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=104550)

# Fitting a multinomial gradient boosting classifier.

gb_multi_cls_params = {'n_estimators':1000, 'max_depth':6, 'learning_rate':0.1}
gb_multi_cls = GradientBoostingClassifier(**gb_multi_cls_params)
gb_multi_cls.fit(X_train, y_train)

# Displaying variable importance.

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

# Computing prediction accuracy on testing set.

y_pred=gb_multi_cls.predict(X_test)
y_test=pd.DataFrame(y_test,columns=['concussion'])
y_pred=pd.DataFrame(y_pred,columns=['predicted'])
df=pd.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['concussion'][i]==df['predicted'][i]:
        match.append(1)
    else:
```

```
        match.append(0)

accuracy=sum(match)/len(match)

print(accuracy)
```

```
          var_name  loss_reduction
3         prevconc        0.507157
2         position        0.411563
0              age        0.057314
1   nyearsplaying        0.023967
0.8571428571428571
```

R Code and Outputs

```
library(xgboost)

concussion_data =
read.csv("C:/Users/coryg/OneDrive/Desktop/STAT_574_Data_Mining/concussions_data.c
sv",
header=T, sep=",")

# Splitting data into 80% training and 20% testing.

sample = sample(c(T,F), nrow(concussion_data), replace=T,
prob=c(0.8, 0.2))
train = concussion_data[sample,]
test = concussion_data[!sample,]

train_x = data.matrix(train[-4])
train_y = data.matrix(train[4])
test_x = data.matrix(test[-4])
test_y = data.matrix(test[4])

# Fitting gradient boosting multinomial classifier.

xgb_multi = xgboost(data=train_x, label=train_y, max.depth=6,
eta=0.1, subsample=0.8, colsample_bytree=0.5,
nrounds=1000, num_class=4, objective="multi:softprob")
```

```r
# Displaying feature importance.

print(xgb.importance(colnames(train_x), model=xgb_multi))

# Computing prediction accuracy for testing data.

pred_prob = predict(xgb_multi, test_x, reshape=T)
pred_prob = as.data.frame(pred_prob)
colnames(pred_prob) <- 0:3
pred_class = apply(pred_prob, 1, function(x)
colnames(pred_prob)[which.max(x)])

match = c()
n = length(test_y)
for (i in 1:n) {
    match[i] = ifelse(pred_class[i]==as.character(test_y[i]), 1, 0)
}

print(accuracy <- mean(match))
```

```
          Feature      Gain      Cover  Frequency
           <char>     <num>      <num>      <num>
1:     concussion 0.3892483 0.08147182 0.05093964
2:       position 0.2357963 0.08299694 0.07570944
3:            age 0.1999811 0.42429226 0.44908729
4:  nyearsplaying 0.1749743 0.41123898 0.42426364
[1] 0.8469388
```