FINAL PROJECT

STAT 702: DATA MINING

SUMUKH SAGAR MANJUNATH

Pima Indians Diabetes Data Set

1. Introduction to Dataset

1.a. About the data

The Pima Indians Diabetes Dataset is a dataset consisting of the results of the tests performed on the people from Pima (a population residing near Phoenix, Arizona, USA) Indian origin. This database is currently owned by National Institute of Diabetes and Digestive and Kidney Diseases and was donated on 9 May 1990.

Several constraints were placed on the selection this dataset from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care).

1.b. Data Summary

There are total of 768 instances in this dataset. Given below are the Features/ Variables of the data:

- **No.Preg**: Number of times pregnant
- **Pl.Gl.Conc**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- **BP**: Diastolic blood pressure (mm Hg)
- **Tric.Thick**: Triceps skin fold thickness (mm)
- **Insulin**: 2-Hour serum insulin (mu U/ml)
- **BMI**: Body mass index (weight in kg/(height in m)^2)
- **Dia.Pedig.Func**: Diabetes pedigree function
- **Age**: Age (years)
- **is.positive**: Class variable (0 or 1)

Class Distribution of Data: (class value 1 is interpreted as "tested positive for diabetes")

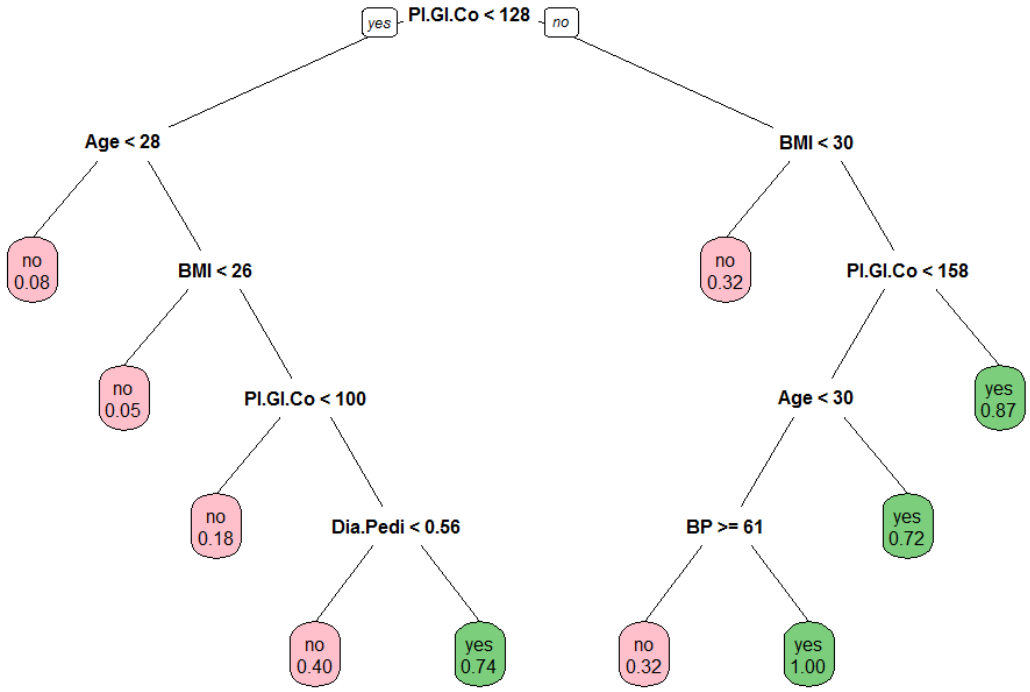
Class Value	Number of instances
0	500
1	268

1.c. Objective

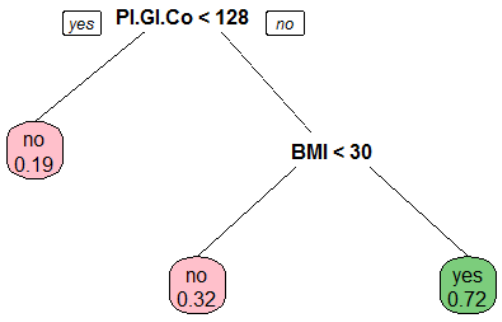
To classify the individuals as positively diagnosed or negatively diagnosed of suffering with diabetes using the classification algorithms like Classification And Regression Trees (CART), Bootstrap AGGregation (BAGGing), Random Forest and compare / comment on the results given out by each of the method.

2. Classification using CART

Let us apply CART using Cross-Validation method without incorporating any loss matrix into the algorithm and not considering the 1-SE rule while pruning the tree. Instead we use the least Cross-Validation error to select our optimal tree. CP value for this subtree is *0.0130597*.



Now we apply CART using Cross-Validation method, not accounting for any loss matrix, but this time we shall prune it using the 1SE rule. This will yield in CP value of *0.01741294*. Three Important variables considered while splitting are Plasma Glucose Concentration (Pl.Gl.Conc), Body Mass Index (BMI) and Age.

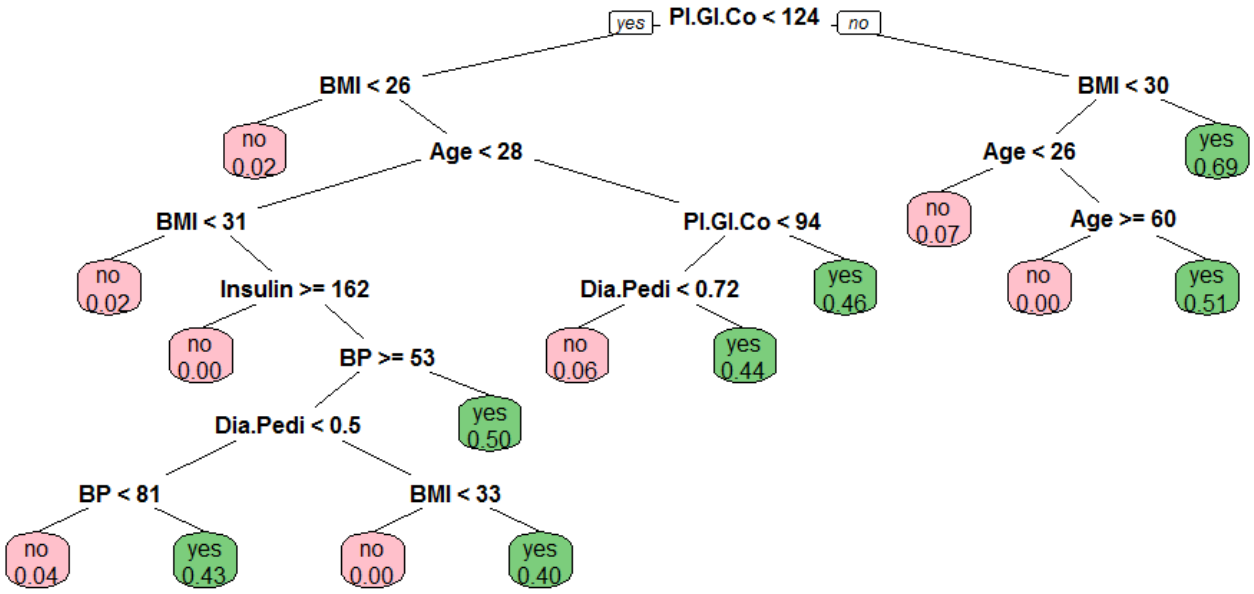


CART Without loss matrix , 1SE

	Pred: Positive	Pred: Negative	
Positive	150	118	268
Negative	57	443	500
	207	561	**768**

False Pos rate	11.40%
False Neg rate	44.02%
Total misclass Rate	**22.78%**

Results from the above settings are good and looks promising because it gives a very simple classifier with a decent Total Misclassification Rate. However the false negative rates are too high which means there will be many cases where people who are actually suffering from diabetes are not diagnosed positively. This is dangerous. So we penalize the classifier for false negative classifications using the loss matrix. We use the following loss matrix: `matrix(c(0,1,4,0), nrow=2, byrow=T),` and follow the 1-SE rule for pruning the tree.



Variable Importance		
Pl.Gl.Conc	**BMI**	**Age**
74.47	60.97	43.32

CART With loss matrix

	Pred: Positive	Pred: Negative	
Positive	259	9	268
Negative	187	313	500
	446	322	**768**

False Pos rate	37.40%
False Neg rate	3.36%
Total misclass Rate	**25.52%**

After application of cost matrix, we observe a drastic change in the false negative misclassification at a small cost of overall misclassification cost. Except few exceptions this trade-off is highly recommended.

3. Classification using BAGGing

Now let us look at another classification method called Bootstrap AGGregation (BAGGing) which embraces Perturb and Combine philosophy to achieve low variance by aggregating multiple results from a low bias method i.e CART. We choose 500 Bootstrap samples to grow 500 trees and run the algorithm.

Among 768 observations in the data, 75% (576) of them are used as the training data used to grow the tree and the rest 25% (192) of them are used as testing data to measure the correctness of the predictions.

Out Of Bag (OOB) Error is defined as misclassifications as measured by sending down the "unused samples" through the tree grown from Bootstrap samples. OOB for this data is 23.70 %.

Variable Importance		
Pl.Gl.Conc	**BMI**	**Age**
50.13	17.05	11.89

Bagging with 500 samples : test data = 192

	Pred		
	Positive	**Negative**	
Positive	45	19	64
Negative	17	111	128
	62	130	**192**

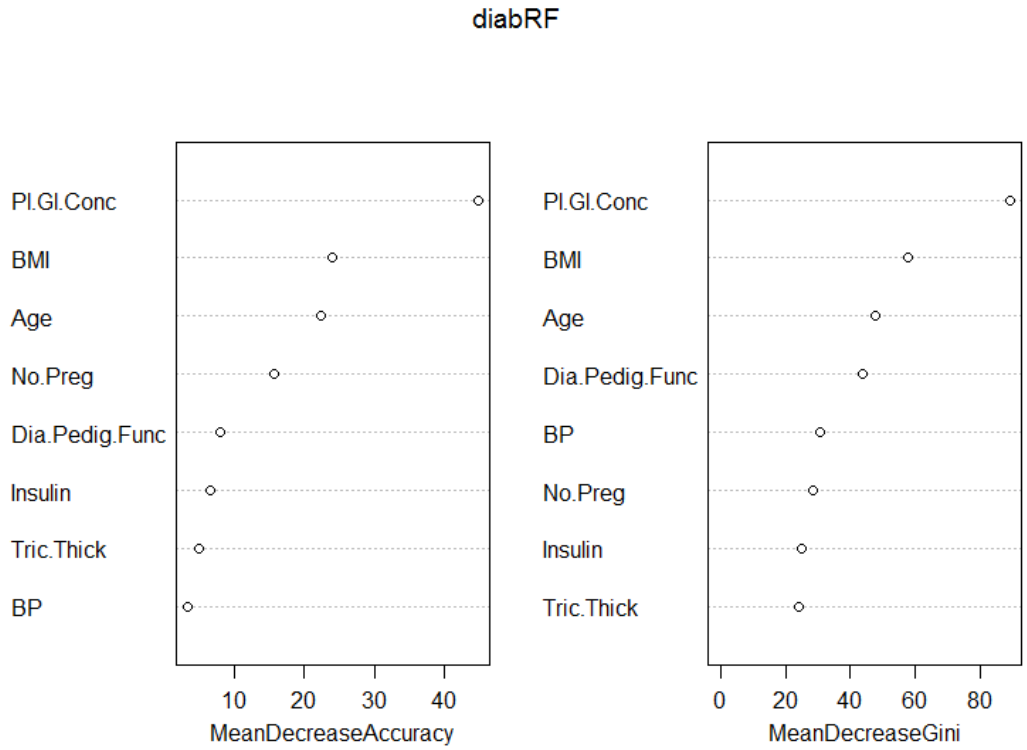
False Pos rate	26.56%
False Neg rate	14.84%
Total misclass Rate	**18.75%**

4. Classification using Random Forest

Lastly, let us look at the Random Forest which is very similar to Bagging except that at each tree split, a random sample of m features is drawn, and only those m features are considered for splitting. Random Forest tries to improve on bagging by de-correlating the trees. Here we run Random Forest using 500 trees and note the results.

	no	yes	MeanDecreaseAccuracy	MeanDecreaseGini
No.Preg	17.40	1.47	15.77	28.69
Pl.Gl.Conc	34.27	34.60	44.72	89.07
BP	4.94	-0.89	3.36	30.52
Tric.Thick	4.15	2.24	4.88	24.06
Insulin	7.19	1.05	6.52	24.88
BMI	15.87	18.36	24.03	57.71
Dia.Pedig.Func	6.20	4.63	7.99	43.59
Age	19.32	9.15	22.45	47.62

Plot showing variable Importance:



Random Forest with 500 trees

	Pred Positive	Pred Negative	
Positive	162	106	268
Negative	71	429	500
	233	535	**768**

False Pos rate	39.55%
False Neg rate	14.20%
Total misclass Rate	**23.04%**

5. Conclusions

- Top three important variables in all three methods are *Plasma Glucose Concentration (Pl.Gl.Conc)*, *Body Mass Index (BMI)* and *Age* respectively.
- Keeping the overall misclassification rate as the factor, Bagging has the least misclassification error among the three methods (CART and BAGGing).
- In this case, false negative error, meaning: diagnosing a person as non-diabetic, when in fact person is suffering from diabetes, is more severe. Therefore we want to reduce the false negative error rate as much as possible.
- Considering the severity of the false negative errors, we might want to go with the CART using loss matrix though overall misclassification rate is slightly higher.
- We might also want to choose CART for this kind classification problem because there is no direct way of incorporating loss matrix in Bagging or Random Forest.

6. Points to ponder

- Since the data is limited to PIMA Indian females of age at least 21, the classifier might not be effective on people from other part of the world, or even male counterparts of the same community.
- Some of the important factors which might cause diabetes like diabetic history of the subject's parents are not taken into account, so causal inferences based wholly on this data might be misleading.

7. References

1. http://mlr.cs.umass.edu/ml/datasets/Pima+Indians+Diabetes.
2. http://www.inside-r.org/packages/cran/adabag/docs/predict.bagging
3. Classification and Regression Trees by Breiman, Friedman, Olshen, and Stone (1984), Chapman & Hall/CRC.
4. http://jessica2.msri.org/attachments/10778/10778-boost.pdf
5. Lecture Notes.

Appendix 1 - Code snippets

```
##########################################################
#         CART TO CLASSIFY CASES OF DIABETES           #
##########################################################

# Using CART with and without 1SE rule to classify positive/Negative
# diabetes outcomes based rest of the predictors

library(rpart)
library(rpart.plot)

# Read in the data in CSV
diab.data = read.csv('pima-indians-diabetes.data', header = F)
colnames(diab.data) = c("No.Preg","Pl.Gl.Conc","BP","Tric.Thick"
              ,"Insulin","BMI","Dia.Pedig.Func","Age","is.positive")
diab.data = as.data.frame(diab.data)

# convert the output variable to factor type
diab.data$is.positive = factor(diab.data$is.positive, levels=0:1,
                               labels=c("no", "yes"))
is.factor(diab.data$is.positive)

head(diab.data)

set.seed(1001)

# Specify the control for RPART with number of cross-validations
my.control <- rpart.control(xval=10, cp=0)

#---------------------------------------------------------
#     Large tree is grown using the loss matrix to
#     penalize the false negative misclassifications
#---------------------------------------------------------
lmat = matrix(c(0,1,4,0), nrow=2, byrow=T)
cfit1 = rpart(formula = is.positive ~ ., data = diab.data,
      method = "class", control = my.control, parms=list(loss=lmat))
#---------------------------------------------------------
#     This large tree is grown using no loss matrix
#---------------------------------------------------------
# Grow alarge tree
cfit1 = rpart(formula = is.positive ~ ., data = diab.data,
              method = "class", control = my.control)


unpruned.tree = printcp(cfit1)
unpruned.tree = as.data.frame(unpruned.tree)

# Find the optimal Complexity Parameter of a subtree based on
# 1SE and minimum crossvalidation error
min.xerr = min(unpruned.tree$xerror)
one.SErr = min(unpruned.tree$xerror) +
  unpruned.tree$xstd[unpruned.tree$xerror == min(unpruned.tree$xerror)]
# optim_cp = max(unpruned.tree$CP[unpruned.tree$xerror < one.SErr])
optim_cp = max(unpruned.tree$CP[unpruned.tree$xerror == min.xerr])
```

```
# Prune bases on the optimal CP found above
pruned.tree = prune(cfit1,cp = optim_cp)
prp(pruned.tree, extra=6, box.col=c("pink", "palegreen3")[pruned.tree$frame$yval])
text(pruned.tree)

# Plot to compare between cross-validation error and training error
numsplits <- cfit1$cptable[, 2]
trainerror <- cfit1$cptable[, 3]
xerror <- cfit1$cptable[, 4]
xstd <- cfit1$cptable[, 5]

plot(numsplits, trainerror, ylim=c(0.2, 1.0), type="l")
lines(numsplits, xerror, lty=2)
lines(numsplits, xerror-xstd, lty=2)
lines(numsplits, xerror+xstd, lty=2)
title("Cross-validation Error Estimates and Training Error")
legend(.02, .4, c("trainerror", "xerror"), lty=c(1,2))

# Calculate the misclassification error, False Positive
# and False negative error
pred.diab = predict(pruned.tree, type="vector") - 1
obsv.diab = as.numeric(diab.data$is.positive) - 1
diab.misclass = dim(diab.data[(pred.diab != obsv.diab),])[1]
diab.misclass
total_obs = dim(diab.data)[1]
total_obs
missclass_rate = (diab.misclass / total_obs)* 100
missclass_rate
false_positive = dim(diab.data[pred.diab == 1 & obsv.diab == 0,])[1]
false_positive
false_negative = dim(diab.data[pred.diab == 0 & obsv.diab == 1,])[1]
false_negative
class.yes = dim(diab.data[diab.data$is.positive == "yes",])[1]
class.yes
class.no = dim(diab.data[diab.data$is.positive == "no",])[1]
class.no
false_positive_rate = false_positive / class.no
false_positive_rate
false_negative_rate = false_negative / class.yes
false_negative_rate


#########################################################
#         BAGGING TO CLASSIFY CASES OF DIABETES        #
#########################################################

# Using Bootstrap AGGregation (BAGGING) with 500 iterations to classify
# positive/Negative diabetes outcomes based on rest of the predictors

detach("package:ipred", unload=TRUE)
library(adabag)
set.seed(1001)

# perform bagging with 500 bootstrap samples
diab.bag = bagging(is.positive ~ ., data= diab.data ,mfinal=500)
```

```
# Derive variable importance
three_bagVar = sort(diab.bag$importance, decreasing = TRUE)[1:3]
three_bagVar

detach("package:adabag", unload=TRUE)
library(ipred)
set.seed(1001)
diab.bag1 = bagging(is.positive ~ ., data= diab.data ,nbag=500, coob=TRUE)

# Calculate the out of bag errors
oob_errBag = diab.bag1$err
oob_errBag


# Calculate misclassification error
smp_size<-floor(.75*nrow(diab.data))
set.seed(1001)
train_ind<-sample(seq_len(nrow(diab.data)),size=smp_size)
diabBag.train<-diab.data[train_ind,]
diabBag.test<-diab.data[-train_ind,]
dim(diabBag.test)
dim(diabBag.train)

diabBag.model <- bagging(is.positive ~ ., data= diabBag.train ,
                         nbag=500, coob=TRUE) # use adabag

diabBag.predc<-predict(diabBag.model,newdata=diabBag.test)

table(diabBag.predc,diabBag.test$is.positive)



##########################################################
#   RANDOM FOREST TO CLASSIFY CASES OF DIABETES        #
##########################################################

library(randomForest)

set.seed(1001)

diabRF = randomForest(is.positive ~ ., data= diab.data,
                      importance=TRUE, ntree=500)

var_importance = as.data.frame(round(importance(diabRF), 2))
var_importance

a = as.numeric(var_importance$MeanDecreaseAccuracy)
names(a) = row.names(var_importance)
three_imp_var_RF = sort(a, decreasing = T)[1:3]
three_imp_var_RF

diabRF.oob = diabRF$err.rate[500,1]
diabRF.oob

varImpPlot(diabRF)
```