STAT 6340 Statistical Machine Learning

Mini Project 6

Author:
Suarez Rodes, Randy

Supervisor:
Choudhary, Pankaj Ph.D.

May 3, 2021

1 Answers

Question 1.a First we grow a full tree with all the data. We present the subsequent splits on Figure 1.

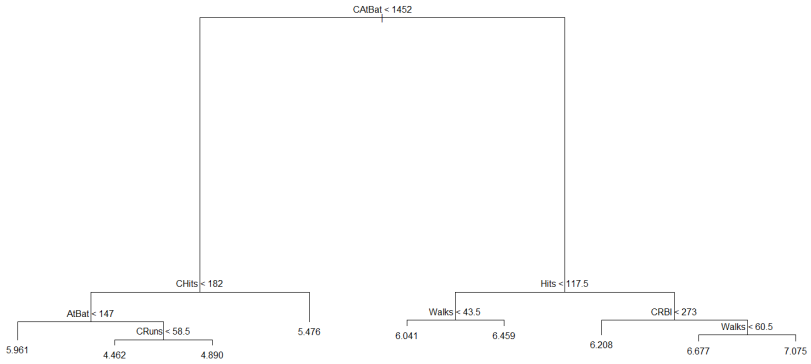

Figure 1: Full Decision Tree

We can see the tree partitions the sample space in the following 9 Regions:

$R_1 = \{X|CAtBat < 1452, CHits < 182, AtBat < 147\}$
$R_2 = \{X|CAtBat < 1452, CHits < 182, AtBat >= 147, CRuns < 58.5\}$
$R_3 = \{X|CAtBat < 1452, CHits < 182, AtBat >= 147, CRuns >= 58.5\}$
$R_4 = \{X|CAtBat < 1452, CHits >= 182\}$
$R_5 = \{X|CAtBat >= 1452, Hits < 117.5, Walks < 43.5\}$
$R_6 = \{X|CAtBat >= 1452, Hits < 117.5, Walks >= 43.5\}$
$R_7 = \{X|CAtBat >= 1452, Hits >= 117.5, CRBI < 273\}$
$R_8 = \{X|CAtBat >= 1452, Hits >= 117.5, CRBI >= 273, Walks < 60.5\}$
$R_9 = \{X|CAtBat >= 1452, Hits >= 117.5, CRBI >= 273, Walks >= 60.5\}$

Using LOOCV we obtained an estimated MSE of 0.2545162.

Question 1.b We performed LOOCV to determine if pruning the tree would be beneficial. We present a comparison of MSE vs Tree Size on Figure 2. We can appreciate that the unpruned tree offers the best results, while the best pruned tree has size 8 and an estimated MSE of 0.2939653. Pruning is not helpful so the best tree is the same as in 1.a.

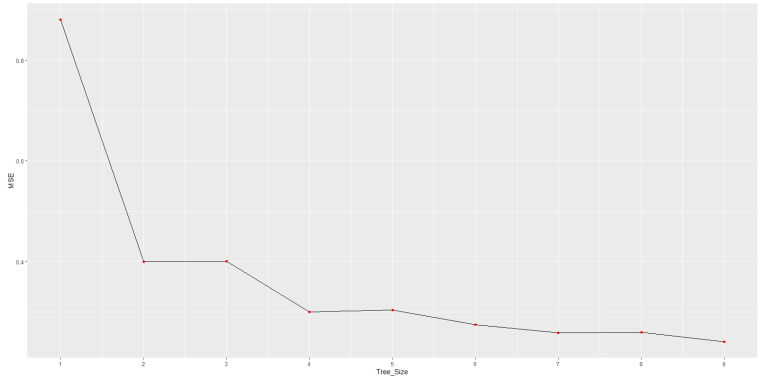

Figure 2: Trees Performance Comparison

For our tree CAtBat (career times at bat) is the most important factor in determining salary, and players with less than 1452 historical times at bat have lower salaries than players over these amount. Given that a player has less than 1452 times at bat, the number of career hits that he made does not seem to have a great impact on the salary. At this same branch, a player with less than 147 times at bat last year earn more salary than those with more than

147 times at bat. For players with more than 1452 historical times at bat, the number of hits is not quite important determining the salary, although, from the players who have more than 1452 times at bat, the more the Walks of last year and the more the Career RBI the greater the salary earned. Therefore we consider the most important predictors are: CAtBat, AtBat, CRBI and Walks.

Question 1.c We fit a bagging model with 1000 bootstrap samples. We can see on Table 1 and Figure 3 the importance of the predictors. Note that historical career stats like CAtBat, CRuns, CRBI and CHits are the most important predictors to determine the salary. Using LOOCV we obtained an estimated MSE of 0.1881713.

Question 1.d We fit a random forest model with 1000 bootstrap samples and number of predictors per tree m = 6. We can see on Table 2 and Figure 4 the importance of the predictors. Once again we have the career stats as the most influential predictors to determine salary. Using LOOCV we obtained an estimated MSE of 0.1802034

Question 1.e We fit a boosting model with 1000 bootstrap samples, with a tree depth d = 1 and with a shrinkage parameter $\lambda = 0.01$ We can see on Table 3 and Figure 5 the influence of the predictors. Similarly to bagging and random forest, the career stats are the higher influence predictors. Using LOOCV we obtained an estimated MSE of 0.2219806.

	%IncMSE	IncNodePurity
AtBat	0.0423	8.5818
Hits	0.0376	7.7941
HmRun	0.0067	2.1102
Runs	0.0153	3.9275
RBI	0.0120	5.7364
Walks	0.0250	7.2763
Years	0.0157	2.2172
CAtBat	0.3426	82.2840
CHits	0.1358	23.6218
CHmRun	0.0135	4.0721
CRuns	0.1824	32.8245
CRBI	0.1674	10.7294
CWalks	0.0181	5.4605
League	-0.0005	0.2028
Division	0.0001	0.2961
PutOuts	0.0041	3.7924
Assists	-0.0006	1.7434
Errors	0.0013	1.7886
NewLeague	-0.0005	0.3797

Table 1: Bagging Predictor Importance

	%IncMSE	IncNodePurity
AtBat	0.0333	7.9761
Hits	0.0353	8.0161
HmRun	0.0084	2.8563
Runs	0.0165	4.6484
RBI	0.0187	6.0675
Walks	0.0200	6.1030
Years	0.0208	7.4679
CAtBat	0.1934	40.8928
CHits	0.1918	34.1012
CHmRun	0.0223	6.2675
CRuns	0.1707	32.1255
CRBI	0.1321	19.4945
CWalks	0.0640	19.9380
League	-0.0000	0.2505
Division	-0.0001	0.2664
PutOuts	0.0024	3.4285
Assists	0.0003	1.8118
Errors	0.0014	1.6870
NewLeague	0.0004	0.3665

Table 2: Random Forest Predictor Importance

var	rel.inf
CAtBat	20.1459
CRuns	16.3605
CHits	12.4726
CRBI	11.2272
CWalks	6.4094
Years	6.0075
CHmRun	4.9388
Hits	4.4498
Walks	4.3957
RBI	3.7450
PutOuts	3.2193
HmRun	2.3677
Runs	1.4490
AtBat	0.8655
Errors	0.8074
Division	0.5169
Assists	0.2601
League	0.2339
NewLeague	0.1278

Table 3: Boosting Predictor Importance

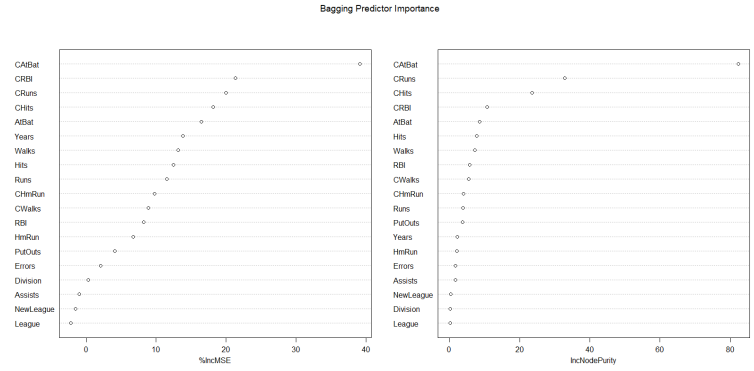

Figure 3: Bagging Predictors Importance

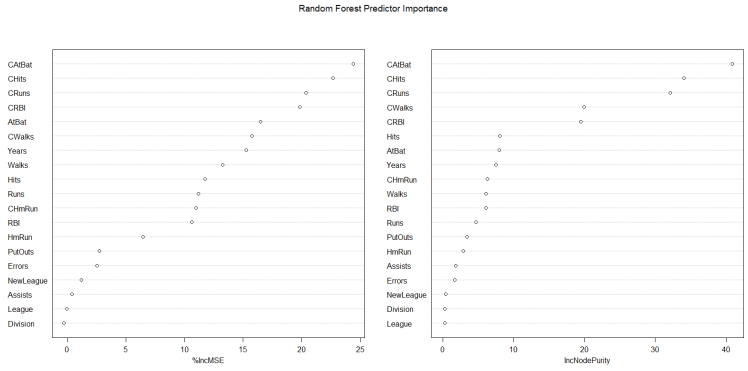

Figure 4: Random Forest Predictors Importance

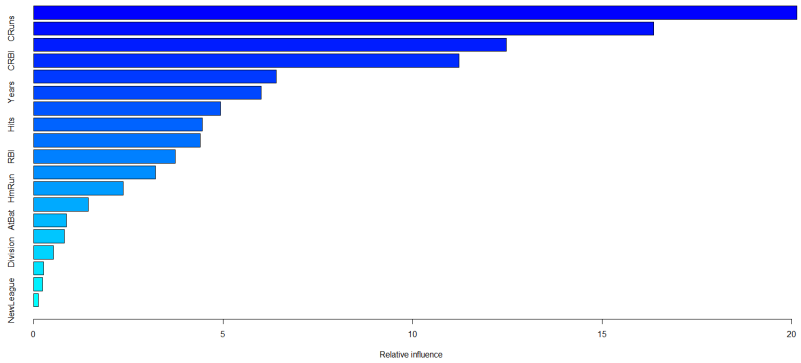

Figure 5: Boosting Predictors Importance

Model	MSE
DT	0.2545
Bagging	0.1882
RF	0.1802
Boosting	0.2220

Table 4: LOOCV Results

Model	MSE
Full	0.421406
PCR	0.410408
PLS	0.413487
RidgeReg	0.408353

Table 5: Project 5 results

Question 1.f We present on Table 4 a summary of the results of LOOCV for the different models. Clearly Random Forest is the best model having the least MSE. On Table 5 we present the results from the previous project. On that occasion ridge regression was our model of choice, although this time this model is beaten by any of our tree models. We recommend tree models to determine the salary of players, specifically random forest is the best course of action for this purpose.

Question 2.a We fitted a support vector classifier using 10 fold CV and selecting the cost from the values: 0.001, 0.01, 0.1, 1-10 and 100. The best model has a cost of 1 with 1053 support vectors (526 for class 0, 527 for class 1). The estimated misclassification error rate is 22.7%.

Question 2.b We fitted a support vector machine with a polynomial kernel of degree two using 10 fold CV and selecting the cost from the values: 0.001, 0.01, 0.1, 1-10 and 100. The best model has a cost of 5 with 1221 support vectors (604 for class 0, 617 for class 1). The estimated misclassification error rate is 26.25%.

Question 2.c We fitted a support vector machine with a radial kernel using 10 fold CV and selecting the cost from the values: 0.001, 0.01, 0.1, 1-10 and 100 and selecting gamma from the values 0.1, 0.5, 1, 2, 3, 4, and 5. The best model has a cost of 6 and gamma = 1 with 688 support vectors (257 for class 0, 431 for class 1). The estimated misclassification error rate is 0.6%.

Question 2.d We present on Table 6 a summary of our results and on Tables 7 and 8 the results of Projects 3 and 4 respectively in order to stablish comparisons. The decision is highly non linear, therefore it is necessary the use of a model with enough flexibility to handle the classification. An svm model with radial kernel and the KNN algorithm are suited for this task. On project 3 the best result was obtained for a KNN model of K = 1, although with this value of K this model may be overfitted. In conclusion we consider an svm with radial kernel is the best model to handle the diabetes data classification.

Classifier	Error
SVC	0.2270
SVMP	0.2625
SVMR	0.0060

Table 6: 10-fold CV Results

Classifier	Error
Full LogReg	0.2195
LDA	0.2230
QDA	0.2445
KNN	0.0015

Table 7: Project 3 LOOCV Results

Classifier	Error
Full LogReg	0.22198
Subset, Forward & Backward	0.22098
RidgeReg	0.22000
Lasso	0.22100

Table 8: Project 4 10-fold CV Results

2 Code

```
#Packages
library(ggplot2) #Used for graphics and visual representations
library(tree) #Used for decision trees
library(randomForest) #Used for bagging and random forest
library(gbm) #Used for boosting
library(e1071) #Used for hyperparameter tuning
library(ISLR)

rdseed=8466 #Seed to replicate results


#Experiment 1
print("Experiment 1")

#Importing data and removing NA values
Hitters=na.omit(Hitters)

#Number of observations
n=nrow(Hitters)
#Number of predictors
p=ncol(Hitters)-1
#Number of bootstrap samples
B=1000

#Creating dataframe with log(Salary)
new_hit=Hitters
new_hit$logSal=log(Hitters$Salary)
new_hit$Salary=NULL

#Dataframe to track errors of each model
error.df=data.frame(MSE=rep(0,4))
row.names(error.df)=c("DT","Bagging","RF","Boosting")


####Question 1.a####

#First we grow a full tree with all the data
hitree=tree(logSal ~ ., new_hit)

summary(hitree)

#Regions
hitree

#Visualizing the tree
plot(hitree)
text(hitree, pretty = 0)

#LOOCV to estimate MSE
hit.errors=sapply(1:n, function(i){

  ti = tree(logSal ~ ., new_hit[-i,])

  ti.pred= predict(ti, new_hit[i,])

  (ti.pred-new_hit$logSal[i])^2
})

#Estimated MSE
error.df["DT","MSE"]=mean(hit.errors)
```

```
print(error.df["DT","MSE"])

####Question 1.b####

#Using LOOCV(K=n) to find performance of the pruned trees
set.seed(rdseed)
cv.hitree = cv.tree(hitree, K=n)

#Best Size = 9, pruning does not help
best_size=which.min(cv.hitree$size)

#Second best (pruned size = 8) tree MSE:
print(cv.hitree$dev[2]/n)

#Plotting Test MSE vs Size
ggplot(data.frame(Tree_Size = cv.hitree$size, MSE= cv.hitree$dev/n ),aes(x=Tree_Size, y=MSE))+
  geom_line()+geom_point(color="red")+scale_x_continuous(breaks=1:9)



#Since the unpruned tree is the best we have the same test MSE as before.
print(error.df["DT","MSE"])


####Question 1.c####

#Fitting a bagging model
set.seed(rdseed)
bag.hit<- randomForest(logSal ~ ., data = new_hit,
                           mtry = p, ntree = B, importance = TRUE)

#Importance of predictors
print(bag.hit$importance)
varImpPlot(bag.hit, main="Bagging Predictor Importance")
#Career predictors like CAtBat, CRuns, CRBI and CHits are the most important

#LOOCV to estimate MSE
bag.errors=sapply(1:n, function(i){

  ti = bag.hit<- randomForest(logSal ~ ., data = new_hit[-i,],
                              mtry = p, ntree = B)

  ti.pred= predict(ti, new_hit[i,])

  (ti.pred-new_hit$logSal[i])^2
})

error.df["Bagging","MSE"]=mean(bag.errors)

print(error.df["Bagging","MSE"])

####Question 1.d####

#Number of predictors sampled per tree
m=round(p/3)

#Fitting a random forest model
set.seed(rdseed)
rf.hit <- randomForest(logSal ~ ., data = new_hit,
                         mtry = m, ntree = B, importance = TRUE)

#Importance of predictors
```

```
print(rf.hit$importance)
varImpPlot(rf.hit, main="Random Forest Predictor Importance")
#Career predictors are the most important predictors

#LOOCV to estimate MSE
rf.errors=sapply(1:n, function(i){

  ti = randomForest(logSal ~ ., data = new_hit[-i,],
                    mtry = m, ntree = B)

  ti.pred= predict(ti, new_hit[i,])

  (ti.pred-new_hit$logSal[i])^2
})

error.df["RF","MSE"]=mean(rf.errors)

print(error.df["RF","MSE"])

####Question 1.e####

#Depth of the trees
d=1

#Shrinkage value
lambda=0.01

#Fitting a boosting model
set.seed(rdseed)
boost.hit <- gbm(logSal ~ ., data = new_hit, distribution = "gaussian",
                    n.trees = B, interaction.depth = d,shrinkage = lambda)

#Predictors' Influence
print(summary(boost.hit))

#Visualizing the effect of some predictors
par(mfrow = c(1, 2))
plot(boost.hit, i = "CAtBat")
plot(boost.hit, i = "CRuns")

#LOOCV to estimate MSE
boost.errors=sapply(1:n, function(i){

  boost.i=gbm(logSal ~ ., data = new_hit[-i,], distribution = "gaussian",
              n.trees = B, interaction.depth = d,shrinkage = lambda)

  yi <- predict(boost.i, newdata = new_hit[i, ],
                n.trees = B)
  (yi - new_hit$logSal[i])^2
})


error.df["Boosting","MSE"]=mean(boost.errors)
print(error.df["Boosting","MSE"])

####Question 1.e####

print(error.df)


#Experiment 2
print("Experiment 2")
```

```r
#Reading the data
diabetes = read.csv("diabetes.csv")

#Renaming predictors
names(diabetes)=c("Pregnancies", "Glucose", "BloodPressure", "SkinThickness",
                  "Insulin",   "BMI",
                  "DiabetesPedigreeFunction", "Age","Outcome" )

#Converting Outcome to factor
diabetes$Outcome=as.factor(diabetes$Outcome)


#Number of observations
nobs=nrow(diabetes)

#Dataframe to track errors of each model
class.error=data.frame(Error=rep(0,3))
row.names(class.error)=c("SVC","SVMP","SVMR")

#Hyperparameters for the svm models
costs=c(0.001, 0.01, 0.1, c(1:10),100)
gamma_val=c(0.1,0.5, 1, 2, 3, 4,5)


####Question 2.a####

#Tuning a support vector classifier
set.seed(rdseed)
svc.tune=tune(svm, Outcome ~ ., data = diabetes, kernel = "linear",
              ranges = list(cost=costs),scale=TRUE)


#Best model cost = 1
bestmod = svc.tune$best.model
summary(bestmod)
print(bestmod$cost)


#Estimated Error Rate
class.error["SVC","Error"]=svc.tune$best.performance
print(class.error["SVC","Error"])

####Question 2.b####

#Tuning support vector machine with polynomial kernel of degree 2
set.seed(rdseed)
svm2.tune=tune(svm, Outcome ~ ., data = diabetes, kernel = "polynomial", degree=2,
              ranges = list(cost=costs),scale=TRUE)

#Best model cost = 5
bestmod2 = svm2.tune$best.model
summary(bestmod2)
print(bestmod2$cost)

#Estimated Error Rate
class.error["SVMP","Error"]=svm2.tune$best.performance
print(class.error["SVMP","Error"])


####Question 2.c####

#Tuning support vector machine with radial kernel
set.seed(rdseed)
```

```
svmr.tune=tune(svm, Outcome ~ ., data = diabetes, kernel = "radial",
               ranges = list(cost=costs, gamma=gamma_val),scale=TRUE)

#Best model cost = 6, gamma = 1
bestmodr = svmr.tune$best.model
summary(bestmodr)
print(bestmodr$cost)
print(bestmodr$gamma)


#Estimated Error Rate
class.error["SVMR","Error"]=svmr.tune$best.performance
print(class.error["SVMR","Error"])

####Question 2.d####

print(class.error)
```