Data 622 Test 1

Sheryl Piechocki

11/13/2020

Description: Use the dataset you used for HW-1 (Blue/Black)

(A) Run Bagging (ipred package)

(B) Run LOOCV (jacknife) for the same dataset
Find the average of the test metric(s).
Compare (A), (B) above with the results you obtained in HW-1 and write 3 sentences explaining the observed difference.

Load Data

Load the data from a csv file and take a look at a summary of the records.

```
path <- "C:/Users/spiec/Documents/Sheryl/CUNY MSDS/DATA622/Data/hw1data.csv"

loadData <- function(csvfile) { read.csv(csvfile,head=T,sep=',',stringsAsFactors=F) }
tst1data <- loadData(path)
tst1data$Y <- as.factor(tst1data$Y)
tst1data$X <- as.factor(tst1data$X)
tst1data$label <- as.factor(tst1data$label)
summary(tst1data)
```

```
##   X      Y       label
##  5 :6   a:6   BLACK:22
##  19:6   b:6   BLUE :14
##  35:6   c:6
##  51:6   d:6
##  55:6   e:6
##  63:6   f:6
```

Split the data 70/30 into train and test sets

```
set.seed(777)
tst1data$label <- as.integer(ifelse(tst1data$label == "BLACK", 0, 1))
testidx <- sample(1:nrow(tst1data), 0.30*nrow(tst1data), replace=F)
train.data <- tst1data[-testidx,]
test.data <- tst1data[testidx,]
table(train.data$label)
```

```
##
##  0  1
## 16 10
```

```
summary(train.data)
```

```
##   X     Y         label
##  5 :4   a:3   Min.   :0.0000
##  19:3   b:6   1st Qu.:0.0000
##  35:5   c:4   Median :0.0000
##  51:5   d:6   Mean   :0.3846
##  55:4   e:3   3rd Qu.:1.0000
##  63:5   f:4   Max.   :1.0000
```

```
table(test.data$label)
```

```
##
## 0 1
## 6 4
```

```
summary(test.data)
```

```
##   X     Y         label
##  5 :2   a:3   Min.   :0.0
##  19:3   b:0   1st Qu.:0.0
##  35:1   c:2   Median :0.0
##  51:1   d:0   Mean   :0.4
##  55:2   e:3   3rd Qu.:1.0
##  63:1   f:2   Max.   :1.0
```

I will be focusing on the Naive Bayes and kNN models, since the logistic regression from homework 1 did not result in any statistically significant features.

Base Naive Bayes Model

Run the base Naive Bayes Model to start

```
library(e1071)
nb.trmodel <- naiveBayes(label~., data=train.data)

nb.tr_pred <- predict(nb.trmodel, train.data[,-c(3)], type='raw')
nb.tr_class <- unlist(apply(round(nb.tr_pred), 1, which.max))-1
nb.tr_tbl <- table(train.data[[3]], nb.tr_class)
nb.tr_cfm <- caret::confusionMatrix(nb.tr_tbl)
nb.tr_cfm
```

```
## Confusion Matrix and Statistics
##
##    nb.tr_class
##      0  1
```

```
##   0 16  0
##   1  0 10
##
##                Accuracy : 1
##                  95% CI : (0.8677, 1)
##     No Information Rate : 0.6154
##     P-Value [Acc > NIR] : 3.295e-06
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6154
##          Detection Rate : 0.6154
##    Detection Prevalence : 0.6154
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
tr.nb_acc <- nb.tr_cfm$overall['Accuracy']

nb.ts_pred <- predict(nb.trmodel, test.data[,-c(3)], type='raw')
nb.ts_class <- unlist(apply(round(nb.ts_pred), 1, which.max))-1
nb.ts_tbl <- table(test.data[[3]], nb.ts_class)
nb.cfm.test <-caret::confusionMatrix(nb.ts_tbl)
nb.cfm.test
```

```
## Confusion Matrix and Statistics
##
##    nb.ts_class
##     0 1
##   0 5 1
##   1 2 2
##
##                Accuracy : 0.7
##                  95% CI : (0.3475, 0.9333)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.6496
##
##                   Kappa : 0.3478
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.7143
##             Specificity : 0.6667
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7000
##          Detection Rate : 0.5000
```

```
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.6905
##
##        'Positive' Class : 0
##
```

```
nb_acc <- nb.cfm.test$overall['Accuracy']
```

kNN Base Model

Run a kNN Base Model

```
library(caret)
knn.trmodel <- knn3(as.factor(label) ~., data = train.data, k = 3)

knn.tr_class <- predict(knn.trmodel, newdata = train.data, type = "class")
knn.tr_pred.prob <- predict(knn.trmodel, newdata = train.data, type = "prob")
knn.tr_tbl <- table(train.data[[3]], knn.tr_class)
knn.cfm.tr <-caret::confusionMatrix(knn.tr_tbl)
knn.cfm.tr
```

```
## Confusion Matrix and Statistics
##
##    knn.tr_class
##      0  1
##   0 15  1
##   1  4  6
##
##                Accuracy : 0.8077
##                  95% CI : (0.6065, 0.9345)
##     No Information Rate : 0.7308
##     P-Value [Acc > NIR] : 0.2604
##
##                   Kappa : 0.5695
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.7895
##             Specificity : 0.8571
##          Pos Pred Value : 0.9375
##          Neg Pred Value : 0.6000
##              Prevalence : 0.7308
##          Detection Rate : 0.5769
##    Detection Prevalence : 0.6154
##       Balanced Accuracy : 0.8233
##
##        'Positive' Class : 0
##
```

```
tr.knn_acc <- knn.cfm.tr$overall['Accuracy']

knn.ts_class <- predict(knn.trmodel, newdata = test.data, type = "class")
```

```
knn.ts_pred.prob <- predict(knn.trmodel, newdata = test.data, type = "prob")
knn.ts_tbl <- table(test.data[[3]], knn.ts_class)
knn.cfm.test <-caret::confusionMatrix(knn.ts_tbl)
knn.cfm.test
```

```
## Confusion Matrix and Statistics
##
##    knn.ts_class
##     0 1
##   0 5 1
##   1 1 3
##
##                Accuracy : 0.8
##                  95% CI : (0.4439, 0.9748)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 0.1673
##
##                   Kappa : 0.5833
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8333
##             Specificity : 0.7500
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.7917
##
##        'Positive' Class : 0
##
```

```
knn_acc <- knn.cfm.test$overall['Accuracy']
```

Part A: Bagging

Make use of Dr. Raman's script for bagging using Naive Bayes

```
library('e1071')
sdf<-train.data
srunModel<-function(sdf) {naiveBayes(label~.,data=sdf[sample(1:nrow(sdf),nrow(sdf),replace=T),])}
slapplyrunmodel<-function(x)srunModel(sdf)

system.time(smodels<-lapply(1:100,slapplyrunmodel))
```

```
##    user  system elapsed
##    0.19    0.00    0.19
```

```
sbagging_preds<-lapply(smodels,FUN=function(M,D=test.data[,-c(3)])predict(M,D,type='raw'))

sbagging_cfm<-lapply(sbagging_preds,FUN=function(P,A=test.data[[3]])
{spred_class<-unlist(apply(round(P),1,which.max))-1
  spred_tbl<-table(factor(A, levels = (0:1)),factor(spred_class, levels = (0:1)))
  spred_cfm<-caret::confusionMatrix(spred_tbl)
  spred_cfm
})

sbagging.perf<-as.data.frame(do.call('rbind',lapply(sbagging_cfm,FUN=function(cfm)c(cfm$overall,cfm$byC

sbagging.perf.mean<-apply(sbagging.perf[-c(2,6:7, 15:18)],2,mean,na.rm = TRUE)
sbagging.perf.var<-apply(sbagging.perf[-c(2,6:7, 15:18)],2,sd, na.rm = TRUE)

sbagging.perf.mean <- as.data.frame(sbagging.perf.mean)
sbagging.perf.mean
```

```
##                 sbagging.perf.mean
## Accuracy                 0.6640000
## AccuracyLower            0.3211784
## AccuracyUpper            0.9086285
## AccuracyNull             0.6300000
## Sensitivity              0.7182976
## Specificity              0.6141667
## Pos Pred Value           0.7316667
## Neg Pred Value           0.5625000
## Precision                0.7316667
## Recall                   0.7182976
## F1                       0.7171525
```

Now, perform bagging using the function from the ipred package. This function develops bootstrap aggregated classification trees.

```
set.seed(2)
library(ipred)
ipred.bag <- bagging(as.factor(label)~., data = train.data, nbagg = 100, coob=TRUE)
print(ipred.bag)
```

```
##
## Bagging classification trees with 100 bootstrap replications
##
## Call: bagging.data.frame(formula = as.factor(label) ~ ., data = train.data,
##     nbagg = 100, coob = TRUE)
##
## Out-of-bag estimate of misclassification error:  0.0385
```

```
ipred.pred <- predict(ipred.bag, test.data[,-c(3)])
ipred.tbl<-table(ipred.pred, test.data$label)
ipred.cfm <- caret::confusionMatrix(ipred.tbl)
ipred.cfm
```

```
## Confusion Matrix and Statistics
```

```
##
##
## ipred.pred 0 1
##          0 4 2
##          1 2 2
##
##                Accuracy : 0.6
##                  95% CI : (0.2624, 0.8784)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 0.6331
##
##                   Kappa : 0.1667
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6667
##             Specificity : 0.5000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5000
##              Prevalence : 0.6000
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.5833
##
##        'Positive' Class : 0
##
```

Part B: Leave One Out Cross Validation (LOO-CV)

Make use of Dr. Raman's script for leave one out cv using Naive Bayes

```
N<-nrow(train.data)

scv_df<-do.call('rbind',lapply(1:N,FUN=function(idx,data=train.data) {
  m<-naiveBayes(label~.,data=data[-idx,])
  p<-predict(m,data[idx,-c(3)],type='raw')
  pc<-unlist(apply(round(p),1,which.max))-1
  list(fold=idx,m=m,predicted=pc,actual=data[idx,c(3)])
  }
))

scv_df<-as.data.frame(scv_df)

loocv_tbl<-table(as.numeric(scv_df$actual),as.numeric(scv_df$predicted))
loocv_caret_cfm<-caret::confusionMatrix(loocv_tbl)

tstloocv.perf<-as.data.frame(do.call('cbind',lapply(scv_df$m,FUN=function(m,data=test.data)
{
  v<-predict(m,data[,-c(3)],type='raw')
  lbllist<-unlist(apply(round(v),1,which.max))-1
}
  )))
```

```
np<-ncol(tstloocv.perf)
loo.predclass<-unlist(apply(tstloocv.perf,1,FUN=function(v){ ifelse(sum(v[2:length(v)])/np<0.5,0,1)}))
loocvtbl<-table(test.data[,c(3)],loo.predclass)
loocv_cfm<-caret::confusionMatrix(loocvtbl)
loocv_cfm
```

```
## Confusion Matrix and Statistics
##
##    loo.predclass
##     0 1
##   0 5 1
##   1 2 2
##
##                Accuracy : 0.7
##                  95% CI : (0.3475, 0.9333)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.6496
##
##                   Kappa : 0.3478
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.7143
##             Specificity : 0.6667
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.5000
##              Prevalence : 0.7000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.6905
##
##        'Positive' Class : 0
##
```

Make use of Dr. Raman's script for leave one out cv using kNN

```
N<-nrow(train.data)

kcv_df<-do.call('rbind',lapply(1:N,FUN=function(idx,data=train.data) {
  m<-knn3(as.factor(label) ~., k = 3, data=data[-idx,])
  p<-predict(m,data[idx,-c(3)],type='prob')
  pc<-unlist(apply(round(p),1,which.max))-1
  list(fold=idx,m=m,predicted=pc,actual=data[idx,c(3)])
  }
))

kcv_df<-as.data.frame(kcv_df)

k.loocv_tbl<-table(as.numeric(kcv_df$actual),as.numeric(kcv_df$predicted))
k.loocv_caret_cfm<-caret::confusionMatrix(k.loocv_tbl)

k.tstloocv.perf<-as.data.frame(do.call('cbind',lapply(kcv_df$m,FUN=function(m,data=test.data)
{
```

```
  v<-predict(m,data[,-c(3)],type='prob')
  lbllist<-unlist(apply(round(v),1,which.max))-1
}
  )))

k.np<-ncol(k.tstloocv.perf)
k.loo.predclass<-unlist(apply(k.tstloocv.perf,1,FUN=function(v){ ifelse(sum(v[2:length(v)])/np<0.5,0,1)
k.loocvtbl<-table(test.data[,c(3)],k.loo.predclass)
k.loocv_cfm<-caret::confusionMatrix(k.loocvtbl)
k.loocv_cfm
```

```
## Confusion Matrix and Statistics
##
##    k.loo.predclass
##     0 1
##   0 5 1
##   1 1 3
##
##                Accuracy : 0.8
##                  95% CI : (0.4439, 0.9748)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 0.1673
##
##                   Kappa : 0.5833
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8333
##             Specificity : 0.7500
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6000
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.7917
##
##        'Positive' Class : 0
##
```

Print results for comparison

```
print(paste('Base NB: ', nb_acc))
```

```
## [1] "Base NB:  0.7"
```

```
print(paste('NB Bagging:',sbagging.perf.mean[1,1]))
```

```
## [1] "NB Bagging: 0.664"
```

```
print(paste('NB LOO-CV:',loocv_cfm$overall[1]))
```

```
## [1] "NB LOO-CV: 0.7"
```

```
print(paste('Base kNN:', knn_acc))
```

```
## [1] "Base kNN: 0.8"
```

```
print(paste('ipred Bagging:',ipred.cfm$overall[1]))
```

```
## [1] "ipred Bagging: 0.6"
```

```
print(paste('kNN LOO-CV:', k.loocv_cfm$overall[1]))
```

```
## [1] "kNN LOO-CV: 0.8"
```

Conclusion

The base Naive Bayes model yields an accuracy of 0.7, but bagging on Naive Bayes drops accuracy to 0.66, while LOO-CV on Naive Bayes maintains the 0.7 accuracy from the base model. Similarly, for kNN, the base model and the LOO-CV models have the same accuracy of 0.8. The bagging model using the ipred package has accuracy of 0.6. The bagging models have not increased accuracy. Bias is increased most likely due to the potential replication of some observations in the bagged samples with such a small sample size (i.e. some observations may not be included in the bagged sample and others may be included more than once). Bagged models usually result in decreased variance. If bias increases, variance decreases. The LOO-CV models give the same accuracy as the base models. In this case we are training on all but one observation, iteratively. This method makes use of all of the data and generally leads to decreased bias. With so few observations here we just maintain the accuracy.