MSDS650 Week 7 Supervised Learning Assignment - Nathan Worsham

Supervised learning algorithms solve regression and classification problems by first being given a training set or known set where the labels or values in question–the "right answers"–are already given. When the answers are discrete or categorical then it is a classification problem and when the answers are continuous values then it is a regression problem. The algorithms then attempt to make a model (similar to pattern detection) so that predictions can be made using the model on labels or response values on a new set (Mathworks.com, 2015).
R packages for performing supervised learning:
RtextTools - https://cran.r-project.org/web/packages/RTextTools/RTextTools.pdf
caret - https://cran.r-project.org/web/packages/caret/caret.pdf
rminer - https://cran.r-project.org/web/packages/rminer/rminer.pdf
bgmm - https://cran.r-project.org/web/packages/bgmm/bgmm.pdf
The measures of quality of the learning algorithm you would expect to see would be accuracy of predicting the values in percentage. This can be simulated with cross-validation. For the example I used a combination of exercises. Using the kaggle.com titanic data set idea, where you can submit your predictions, I wanted to try to come at it using the ensemble method from the Breast Cancer Rtexttools exercise. The Rtexttools has a built-in command of `cross-validate` can be used to access the performance of each algorithm. I used this concept to remove algorithms that performed below 0.80–though honestly I'm not sure why in cross-validation results the GLMNET scored 1.587607, as I am not sure how it can perform over 100%. However the first time I did my submission I didn't realize I left all of the algorithms in and actually scored higher (0.78469) than when I removed the 2 lower performing. But the result with the 2 lower performing algorithms removed, it still scored better (0.77990) than the original exercise (0.76555). At this point trying variations of 4 and 5 higher performing algorithms performed the exact same ensemble prediction as the 2 lower removed, which I suppose makes sense because we are looking for an agreement between algorithms.

Example

As mentioned earlier I did my example using the Titanic dataset from kaggle.com but wanted to try doing the submission using the Rtexttools package from the Breast Cancer exercise. I think the thing that interested me most about the kaggle example is first that the exercise was very helpful and clear to understand and the fact that you could upload your results to see how you did. I was floored to see that there were a handful of submission with 100% accuracy, I think this is what hooked me the most. I first recreated the Breast Cancer exercise using only my training data from the kaggle.com titanic set:

```
setwd("/Users/worshamn/Dropbox/Documents/Regis/MSDS650/week7/")
library(RTextTools)
data <- read.csv("train.csv")
#Remove name
data <- data[-4]
#Remove Ticket
data <- data[-8]
#change numeric classes to add text to them since this is a "text tool"
Pclass <- as.vector(apply(as.matrix(data[3], mode="character"),1,paste,"Pclass",sep="",
collapse=""))
Age <- as.vector(apply(as.matrix(data[5], mode="character"),1,paste,"Age",sep="",
collapse=""))
SibSp <- as.vector(apply(as.matrix(data[6], mode="character"),1,paste,"SibSp",sep="",
collapse=""))
Parch <- as.vector(apply(as.matrix(data[7], mode="character"),1,paste,"Parch",sep="",
collapse=""))
Fare <- as.vector(apply(as.matrix(data[8], mode="character"),1,paste,"Fare",sep="",
collapse=""))
#create the training data
training_data <- cbind(data[1],data[2],Pclass,data[4],Age,SibSp,Parch,Fare,data[9],
data[10])
#get the labels
training_codes <- training_data[2]
#remove labels from training data
training_data <- training_data[-2]
#run remaining text tool functions
require(tm)
matrix <- create_matrix(training_data, language="english", removeNumbers=FALSE,
stemWords=FALSE, removePunctuation=FALSE, weighting=weightTfIdf)
container <- create_container(matrix,t(training_codes),trainSize = 1:891,virgin=FALSE)
models <- train_models(container, algorithms=c("MAXENT","SVM","GLMNET","SLDA"
,"TREE","BAGGING","BOOSTING","RF"))
results <- classify_models(container, models)
analytics <- create_analytics(container, results)
analytics@ensemble_summary
analytics@algorithm_summary
analytics@label_summary
```

Now as mentioned above, run cross-validation to see which algorithm performed the best:

```
cross_validate(container, 4, algorithm=c("MAXENT","SVM","GLMNET","SLDA","TREE"
,"BAGGING","BOOSTING","RF"))
SVM <- cross_validate(container, 4, "SVM")
GLMNET <- cross_validate(container, 4, "GLMNET")
MAXENT <- cross_validate(container, 4, "MAXENT")
SLDA <- cross_validate(container, 4, "SLDA")
BAGGING <- cross_validate(container, 4, "BAGGING")
BOOSTING <- cross_validate(container, 4, "BOOSTING")
RF <- cross_validate(container, 4, "RF")
NNET <- cross_validate(container, 4, "NNET")
TREE <- cross_validate(container, 4, "TREE")
performance <- cbind(SVM$meanAccuracy,GLMNET$meanAccuracy,
  MAXENT$meanAccuracy,SLDA$meanAccuracy,BAGGING$meanAccuracy,
  BOOSTING$meanAccuracy,RF$meanAccuracy,NNET$meanAccuracy
  ,TREE$meanAccuracy)
colnames(performance) <- c('SVM','GLMNET','MAXENT','SLDA','BAGGING','BOOSTING'
,'RF','NNET','TREE')
> performance
           SVM   GLMNET     MAXENT      SLDA   BAGGING  BOOSTING        RF      NNET
[1,] 0.8833741 1.587607 0.02673924 0.8454323 0.9240826 0.9406644 0.8849548 0.6159933
          TREE
[1,] 0.7833904
```

Trying the result again with removing the 2 lower performing algorithms:

```
matrix <- create_matrix(training_data, language="english", removeNumbers=FALSE,
  stemWords=FALSE, removePunctuation=FALSE, weighting=weightTfIdf)
container <- create_container(matrix,t(training_codes),trainSize = 1:891,virgin=FALSE)
models <- train_models(container, algorithms=c("SVM","GLMNET","SLDA","BAGGING"
  ,"BOOSTING","RF"))
results <- classify_models(container, models)
analytics <- create_analytics(container, results)
```

Here I would say I was stuck for awhile because the kaggle.com exercise made it easy to apply his model to his test data. I finally figured out I would have to combine my training and test data but only train on the right rows and test on the remaining to get it to run a prediction for me.

```
testdata <- read.csv("test.csv")
library(plyr)
newdata <- rbind.fill(data, testdata)
#Remove Name
newdata <- newdata[-4]
#Remove Ticket
newdata <- newdata[-8]
#change numeric classes to add text to them since this is a "text tool"
Pclass <- as.vector(apply(as.matrix(newdata[3], mode="character"),1,paste,"Pclass",
sep="",collapse=""))
Age <- as.vector(apply(as.matrix(newdata[5], mode="character"),1,paste,"Age",
sep="",collapse=""))
SibSp <- as.vector(apply(as.matrix(newdata[6], mode="character"),1,paste,"SibSp",
sep="",collapse=""))
Parch <- as.vector(apply(as.matrix(newdata[7], mode="character"),1,paste,"Parch",
sep="",collapse=""))
Fare <- as.vector(apply(as.matrix(newdata[8], mode="character"),1,paste,"Fare",
sep="",collapse=""))
#create the training data
new_training_data <- cbind(newdata[1],newdata[2],Pclass,newdata[4],Age,SibSp,Parch,
  Fare,newdata[9],newdata[10])
#get the labels
new_training_codes <- new_training_data[2]
#remove labels from training data
new_training_data <- new_training_data[-2]
require(tm)
matrix <- create_matrix(new_training_data, language="english", removeNumbers=FALSE,
  stemWords=FALSE, removePunctuation=FALSE, weighting=weightTfIdf)
container <- create_container(matrix,t(new_training_codes),trainSize = 1:891,testSize =
  892:1309,virgin=FALSE)
testContainer <- create_container(matrix,t(new_training_codes),testSize = 892:1309,
  virgin=TRUE)
#accidentally left in all algorithms, but received better results 0.78469
models <- train_models(container, algorithms=c("MAXENT","SVM","GLMNET","SLDA",
  "TREE","BAGGING","BOOSTING","RF"))
#0.77990 performance
models <- train_models(container, algorithms=c("SVM","GLMNET","SLDA","BAGGING"
,"BOOSTING","RF"))
#try just top 2 - received same results as previous
models <- train_models(container, algorithms=c("BAGGING","BOOSTING"))
#try adding 3 more performers - still same results
models <- train_models(container, algorithms=c("BAGGING","BOOSTING","TREE","RF"))
# predict
results <- classify_models(testContainer, models)
results
analytics <- create_analytics(testContainer, results)
```

Method I would use over and over to get the `CONSENSUS_CODE`, which I assumed was the ensemble agreement and formatting it to submit to kaggle.com. The problem I ran into here is that it would number my results 1-418 when they needed to look like the original submission from the example that start with numbering at 892. So I elected to combine the column from the first submission with the classification response of my new submission

```
analytics@document_summary$CONSENSUS_CODE
write.csv(analytics@document_summary, "DocumentSummary.csv")
```

```
submission1 <- read.csv("submission.csv")
submission2 <- read.csv("DocumentSummary.csv")
newsubmission <- cbind(submission1,submission2)
newsubmission <- newsubmission[c('PassengerId','x')]
write.table(newsubmission, file = "submission2.csv", col.names = TRUE, row.names = FALSE
  , sep = ",")
```

References

Mathworks.com, 2015. Retrieved from http://www.mathworks.com/discovery/supervised-learning.html