Summary of models trained

Team 1

09/10/2020

Summary of trained model performance

First we load the models.

```
library(plyr)
library(caret)
library(e1071)
rm(list=ls())
load(file = "DataWrangling/Featuresselected.RData")
load(file = "Models/SVMradmodel.RData")
load(file = "Models/svmpolymodel.RData")
load(file = "Models/svmlinmodel.RData")
load(file = "Models/RF.RData")
load(file = "Models/Neural_Network.RData")
load(file = "Models/NB.RData")
load(file = "Models/LR.RData")
load(file = "Models/LDAmodel.RData")
load(file = "Models/knnmodel.RData")
train.df$Cath <- as.factor(ifelse(train.df$Cath == 0,"N","Y"))
test.df$Cath <- as.factor(ifelse(test.df$Cath == 0,"N","Y"))
```

Individual Models

Here is a summary of the performance of our individual models.

k-Nearest Neighbours

```
knn_result <- predict(knn.model, test.df[knn.features])
confusionMatrix(knn_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 14  6
##          Y  4 37
##
##                Accuracy : 0.8361
##                  95% CI : (0.7191, 0.9185)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.01412
##
##                   Kappa : 0.6183
##
##  Mcnemar's Test P-Value : 0.75183
##
##             Sensitivity : 0.7778
##             Specificity : 0.8605
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 0.9024
##              Prevalence : 0.2951
##          Detection Rate : 0.2295
##    Detection Prevalence : 0.3279
##       Balanced Accuracy : 0.8191
##
##        'Positive' Class : N
##
```

Linear Discriminant Analysis

```
lda_result <- predict(lda.model, test.df[lda.features$optVariables])
confusionMatrix(lda_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 14  3
##          Y  4 40
##
##                Accuracy : 0.8852
##                  95% CI : (0.7778, 0.9526)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.0007505
##
##                   Kappa : 0.7196
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.7778
##             Specificity : 0.9302
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.9091
##              Prevalence : 0.2951
##          Detection Rate : 0.2295
##    Detection Prevalence : 0.2787
##       Balanced Accuracy : 0.8540
##
##        'Positive' Class : N
##
```

Logistic Regression

```
lr_result <- as.factor(ifelse(predict(LR_model, test.df[lr.features$optVariables])==1,
                              "Y","N"))
confusionMatrix(lr_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 13  3
##          Y  5 40
##
##                Accuracy : 0.8689
##                  95% CI : (0.7578, 0.9416)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.002264
##
##                   Kappa : 0.6742
##
##  Mcnemar's Test P-Value : 0.723674
##
##             Sensitivity : 0.7222
##             Specificity : 0.9302
##          Pos Pred Value : 0.8125
##          Neg Pred Value : 0.8889
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.2623
##       Balanced Accuracy : 0.8262
##
##        'Positive' Class : N
##
```

Naive Bayes

```
nb_result <- as.factor(ifelse(predict(NB_model, test.df[nb.features$optVariables])==1,
                              "Y","N"))
confusionMatrix(nb_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 13  6
##          Y  5 37
##
##                Accuracy : 0.8197
##                  95% CI : (0.7002, 0.9064)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.02988
##
##                   Kappa : 0.5734
```

```
## 
##  Mcnemar's Test P-Value : 1.00000
## 
##             Sensitivity : 0.7222
##             Specificity : 0.8605
##          Pos Pred Value : 0.6842
##          Neg Pred Value : 0.8810
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.3115
##       Balanced Accuracy : 0.7913
## 
##        'Positive' Class : N
## 
```

Random Forest

```
rf_result <- as.factor(ifelse(predict(RF_model, test.df[rf.features$optVariables])==1,
                              "Y","N"))
confusionMatrix(rf_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  N  Y
##          N 11  1
##          Y  7 42
## 
##                Accuracy : 0.8689
##                  95% CI : (0.7578, 0.9416)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.002264
## 
##                   Kappa : 0.6509
## 
##  Mcnemar's Test P-Value : 0.077100
## 
##             Sensitivity : 0.6111
##             Specificity : 0.9767
##          Pos Pred Value : 0.9167
##          Neg Pred Value : 0.8571
##              Prevalence : 0.2951
##          Detection Rate : 0.1803
##    Detection Prevalence : 0.1967
##       Balanced Accuracy : 0.7939
## 
##        'Positive' Class : N
## 
```

Neural Network

```
nn_result <- as.factor(ifelse(predict(nn1, test.df)==1,"Y","N"))
confusionMatrix(nn_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 13  4
##          Y  5 39
##
##                Accuracy : 0.8525
##                  95% CI : (0.7383, 0.9302)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.005995
##
##                   Kappa : 0.6395
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7222
##             Specificity : 0.9070
##          Pos Pred Value : 0.7647
##          Neg Pred Value : 0.8864
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.2787
##       Balanced Accuracy : 0.8146
##
##        'Positive' Class : N
##
```

Support Vector Machines: Linear kernal

```
svm_lin_result <- predict(svmlin.model, test.df[svmlin.features$optVariables])
confusionMatrix(svm_lin_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 12  2
##          Y  6 41
##
##                Accuracy : 0.8689
##                  95% CI : (0.7578, 0.9416)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.002264
##
##                   Kappa : 0.663
##
##  Mcnemar's Test P-Value : 0.288844
```

```
##
##             Sensitivity : 0.6667
##             Specificity : 0.9535
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.8723
##              Prevalence : 0.2951
##          Detection Rate : 0.1967
##    Detection Prevalence : 0.2295
##       Balanced Accuracy : 0.8101
##
##        'Positive' Class : N
##
```

Support Vector Machines: Polynominal kernal

```
svm_poly_result <- predict(svmpoly.model, test.df[svmpoly.features$optVariables])
confusionMatrix(svm_poly_result,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 13  5
##          Y  5 38
##
##                Accuracy : 0.8361
##                  95% CI : (0.7191, 0.9185)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.01412
##
##                   Kappa : 0.6059
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.7222
##             Specificity : 0.8837
##          Pos Pred Value : 0.7222
##          Neg Pred Value : 0.8837
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.2951
##       Balanced Accuracy : 0.8030
##
##        'Positive' Class : N
##
```

Support Vector Machines: Radial Basis kernal

```
svm_rad_result <- predict(svmrad.model, test.df[svmrad.features$optVariables])
confusionMatrix(svm_rad_result,test.df$Cath)
```

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

```
##
##           Reference
## Prediction  N  Y
##          N 13  2
##          Y  5 41
##
##                Accuracy : 0.8852
##                  95% CI : (0.7778, 0.9526)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.0007505
##
##                   Kappa : 0.7101
##
##  Mcnemar's Test P-Value : 0.4496918
##
##             Sensitivity : 0.7222
##             Specificity : 0.9535
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 0.8913
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.2459
##       Balanced Accuracy : 0.8379
##
##        'Positive' Class : N
##
```

Ensemble Models

We must first define some custom functions. The function vote-ensemble() takes a data set and for every row returns the average of each feature. The function generate_ensemble_df() generates a dataset with features corresponding to the predicted results from the above models, based on their probability. It also aggregates all three SVM features into one weighted predictor, as SVM does not have a probability associated with predictions.

```
pred.df <- generate_ensemble_df(train.df)
```

Ensemble by voting

```
ensem_result_test <- vote_ensemble(generate_ensemble_df(test.df),
                                   label = 'Cath', prob='class',input='prob')
confusionMatrix(ensem_result_test,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 13  3
##          Y  5 40
##
##                Accuracy : 0.8689
##                  95% CI : (0.7578, 0.9416)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.002264
##
##                   Kappa : 0.6742
##
##  Mcnemar's Test P-Value : 0.723674
##
##             Sensitivity : 0.7222
##             Specificity : 0.9302
##          Pos Pred Value : 0.8125
##          Neg Pred Value : 0.8889
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.2623
##       Balanced Accuracy : 0.8262
##
##        'Positive' Class : N
##
```

Training models on the combined data frame.

We can also train models on the new data frame generated above by generate_ensemble_df(). We choose two simple models, as we have few features.

Logistic Regression Ensemble

```
control <- trainControl(method="repeatedcv", number=10)
lr_ensem <- train(Cath ~., data = pred.df, method="glm", family = "binomial",
                  trControl=control)

ensem_lr_test=predict(lr_ensem,generate_ensemble_df(test.df))
confusionMatrix(ensem_lr_test,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 13  5
##          Y  5 38
##
##                Accuracy : 0.8361
##                  95% CI : (0.7191, 0.9185)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.01412
##
##                   Kappa : 0.6059
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.7222
##             Specificity : 0.8837
##          Pos Pred Value : 0.7222
##          Neg Pred Value : 0.8837
##              Prevalence : 0.2951
##          Detection Rate : 0.2131
##    Detection Prevalence : 0.2951
##       Balanced Accuracy : 0.8030
##
##        'Positive' Class : N
##
```

Random Forest Ensemble

```
control <- trainControl(method="repeatedcv", number=10)
rf_ensem<-train(Cath ~., data = pred.df, method="rf", family = "binomial",
                trControl=control)

ensem_rf_test=predict(rf_ensem,generate_ensemble_df(test.df))
confusionMatrix(ensem_rf_test,test.df$Cath)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  N  Y
##          N 11  2
##          Y  7 41
##
##                Accuracy : 0.8525
##                  95% CI : (0.7383, 0.9302)
##     No Information Rate : 0.7049
##     P-Value [Acc > NIR] : 0.005995
##
##                   Kappa : 0.6142
##
##  Mcnemar's Test P-Value : 0.182422
##
##             Sensitivity : 0.6111
##             Specificity : 0.9535
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8542
##              Prevalence : 0.2951
##          Detection Rate : 0.1803
##    Detection Prevalence : 0.2131
##       Balanced Accuracy : 0.7823
##
##        'Positive' Class : N
##
```

We note that all the above ensemble attempts have comparable performance to the individual models.