GroupFinal

Robert West

11/10/2020

Helper functions

Simulation for Methods on Final Project

```
#Read in data
wine = read.csv("winequality-red.csv", sep=";")

#Split data groups based on Quality
split = factor((wine$quality > 5), labels = c("Bad", "Good"))
mean(split=="Good")
```

```
## [1] 0.5347092
```

```
wineTotal = wine %>%
  mutate(split=split)
```

Split data

```
train = sample(1:nrow(wineTotal), size=nrow(wineTotal)*0.8, replace = F)

wineTrain = wineTotal[train, ]%>%dplyr::select(-quality)
wineTest = wineTotal[-train, ]%>%dplyr::select(-quality)
```

Logistic Regression

```
outliers = which(wineTotal$volatile.acidity > 1.2)
outliers = append(outliers, which(wineTotal$chlorides > 0.25))
outliers = append(outliers, which(wineTotal$sulphates > 1.5))

logistic = wineTotal[-outliers,]

# Create a new variable and remove variables
logistic = logistic %>%
  mutate(ratio.sulfur.dioxide = free.sulfur.dioxide/total.sulfur.dioxide) %>%
  dplyr::select(-c(fixed.acidity,free.sulfur.dioxide,total.sulfur.dioxide,citric.acid,density,residual.
```

```
train.logistic = logistic[train,]
test.logistic = logistic[-train,]

#Create the logistic regression model
glm.fit.wine = glm(split ~ volatile.acidity+pH+sulphates+I(sulphates^2)+
                           I(sulphates^3)+alcohol+ratio.sulfur.dioxide,
                   data=logistic, family = binomial)

glm.probs = predict(glm.fit.wine,type='response', newdata = test.logistic)

glm.pred = factor((glm.probs > 0.5), labels = c("Bad", "Good"))

mean(glm.pred != test.logistic$split)
```

```
## [1] 0.25
```

Tree method

```
#Logistic regression and tree methods removed the same observations
treeTrain = wineTrain[-outliers, ]
treeTest = wineTest[-outliers, ]

treeMod= tree(split~., data=treeTrain)

treePred = predict(treeMod, newdata=treeTest, type = "class")

mean(treePred != treeTest$split)
```

```
## [1] 0.3133333
```

Bagged trees

```
bagMod = randomForest(split ~ ., data=treeTrain, mtry=11, importance=T)
bagPred = predict(bagMod, newdata=treeTest, type="class")
mean(bagPred != treeTest$split)
```

```
## [1] 0.1666667
```

GBM

```
control=trainControl(method="cv", number=5, search="grid")

tunegrid=expand.grid(n.trees=1000,
                     interaction.depth=5,
                     shrinkage=c(0.001,0.005,0.01,0.015,0.02, 0.03, 0.05, 0.1),
```

```
                    n.minobsinnode=3)

gb_gridsearch=train(split~., data=treeTrain, method="gbm", metric="Accuracy",
                            tuneGrid=tunegrid, trControl=control, verbose=F)


#Cross Validation over train set which increases the test error
boost.data=gbm(split~., data=treeTrain, distribution="multinomial", n.trees=10000,
               shrinkage=gb_gridsearch$bestTune$shrinkage, interaction.depth=4)

#Prediction
gbmPred = predict.gbm(object = boost.data,
                      newdata = treeTest,
                      n.trees = 500,
                      type = "response")
labels = colnames(gbmPred)[apply(gbmPred, 1, which.max)]

mean(labels != treeTest$split)
```

```
## [1] 0.2166667
```

Random Forest

```
control=trainControl(method="cv", number=5, search="grid")
tunegrid=expand.grid(mtry=c(1:11))

rf_gridsearch=train(split~., data=treeTrain, method="rf", metric="Accuracy",
                            tuneGrid=tunegrid, trControl=control)

rfMod=rf_gridsearch$finalModel

rfPred = predict(rfMod, newdata=treeTest)
mean(rfPred != treeTest$split)
```

```
## [1] 0.17
```

LDA/QDA

```
#Transformations in an attempt to meet Normality assumptions
LDA = wineTotal
LDA$split = wineTotal$split
LDA$volatile.acidity = wineTotal$volatile.acidity
LDA$chlorides = log10(wineTotal$chlorides)
LDA$pH = wineTotal$pH
LDA$sulphates = log10(wineTotal$sulphates)
LDA$alcohol = sqrt(wineTotal$alcohol)
LDA$ratio_sulfur.dioxide = wineTotal$free.sulfur.dioxide /
  wineTotal$total.sulfur.dioxide
LDA = subset(LDA, select = c(volatile.acidity, chlorides,
```

```
                                pH, sulphates, alcohol,
                                ratio_sulfur.dioxide, split))

#LDA
lda.fit = lda(split~. , data = wineTrain)
lda.pred = predict(lda.fit, wineTest)$class
mean(lda.pred != wineTest$split)
```

```
## [1] 0.2375
```

```
#QDA
qda.fit = qda(split~. , data = wineTrain)
qda.pred <- predict(qda.fit, wineTest)$class
mean(qda.pred != wineTest$split)
```

```
## [1] 0.234375
```

KNN Classification

```
#Variables to include after selection performed
preds = c(2,5,6,7,9,10,11)

X_trn = wineTrain[ ,preds]
X_tst  = wineTest[ ,preds]

#Grid over which to search for optimal K
k = c(1, 3, 5, 10, 25, 50, 100)
knn_tst_rmse = sapply(k, make_knn_pred,
                      train_X = X_trn,
                      test_X = X_tst,
                      train_Y = wineTrain$split,
                      test_Y = wineTest$split)

# determine "best" k
best_k = k[which.min(knn_tst_rmse)]

#Build the optimal model
make_knn_pred(k=best_k,
              train_X = X_trn,
              test_X = X_tst,
              train_Y = wineTrain$split,
              test_Y = wineTest$split)
```

```
## [1] 0.296875
```

KNN Regression (Not sure if we need)

```
k = c(1, 3, 5, 10, 25, 50, 100)
knn_tst_rmse = sapply(k, make_knn_pred_Reg,
                      train_X = X_trn,
                      test_X = X_tst,
                      train_Y = wineTotal[train, 12],
                      test_Y = wineTotal[-train, 12])

# determine "best" k
best_k = k[which.min(knn_tst_rmse)]

#Build the optimal model
make_knn_pred_Reg(k=best_k,
                  train_X = X_trn,
                  test_X = X_tst,
                  train_Y = wineTotal[train, 12],
                  test_Y = wineTotal[-train, 12])
```

```
## [1] 0.7503333
```

SVM(Radial)

```
radialSVM = tune(svm, split ~., data=wineTrain, kernel="radial", scale=T,
                  ranges=list(
                    cost=c(0.001, 0.01, 0.1, 1),
                    gamma=c(0.5, 1, 2, 3, 4)
                  ))
radialSVMPred = predict(radialSVM$best.model, newdata=wineTest)
mean(radialSVMPred != wineTest$split)
```

```
## [1] 0.24375
```