Statistical Data Mining

Swapnil Awasthi

Homework 4

Create a new dataset using the same price changes from the past 25 years of S&P Adjusted Closing Prices from Finance.Yahoo.Com.

Add to this the change in interest rates, similarly from the previous 25 years.
Choose a third category (oil, foreign exchange rates, CPI) and include those changes.
This creates 15 columns of data to predict the price change.

Modify this price change to a categorical value for:
Awful (Change <-1 standard deviation)
Bad (-1 stdev <= Change < -.3 stdev)
Unchanged(-.3 stdev <= Change < .3 stdev)
Good (.3 stdev <= Change < 1 stdev)
Great (Change >= 1 stdev)
Model the price change using the three models and determine if any of them perform well.
Determine a reasonable experiment (cross validation, testing/training) and give an executive summary of your findings.
***Link to R code**: https://github.com/swapnilawasthi/sdmhw5/blob/master/hw4_soln.R*

Naïve Bayes

```
> # Modelling using NaiveBayes
> model.NB <- NaiveBayes(priceDir ~ snp_cat:
data=trng.d)
> predictions <- predict(model.NB, test.d)
There were 31 warnings (use warnings() to se
> confusionMatrix(test.d$priceDir, predictic
Confusion Matrix and Statistics

          Reference
Prediction High  Low
      High   18 1205
      Low    24 1328

               Accuracy : 0.5227
                 95% CI : (0.5032, 0.5422)
    No Information Rate : 0.9837
    P-Value [Acc > NIR] : 1

                  Kappa : -0.0032
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.42857
            Specificity : 0.52428
         Pos Pred Value : 0.01472
         Neg Pred Value : 0.98225
             Prevalence : 0.01631
         Detection Rate : 0.00699
   Detection Prevalence : 0.47495
      Balanced Accuracy : 0.47643

       'Positive' Class : High

> |
```

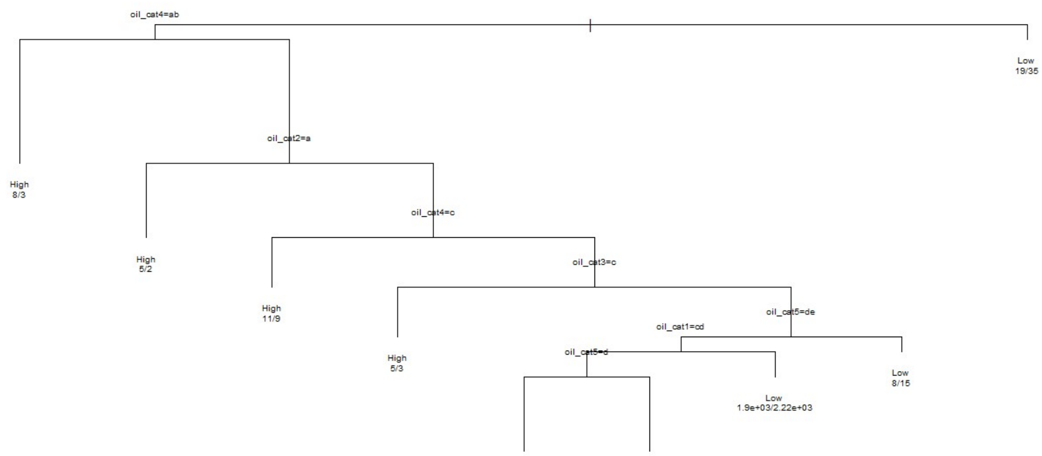
Naïve Bayes confusion matrix

Summary: Our Naïve Bayes model is giving an average accuracy of 52.3% with an 95% confidence that our values will be between .5032 and .5422.
Our true positive rate is .428 and true negative rate is .524, our model is better at predicting proportion of negatives that are correctly identified.

<u>Recursive partition tree</u>

```
> # Modelling using Recursive Partition Tree
> #install.packages('rpart')
> library('rpart')
> model.rpt <- rpart(priceDir ~ snp_cat3+ snp_cat4 + snp_cat5 + bnd_cat
ng.d, cp=0)
> plot(model.rpt)
> text(model.rpt, use.n= T, digits=3, cex=0.6)
> prediction.rpt <- predict(model.rpt, newdata = test.d, type="class")
> printcp(model.rpt)

Classification tree:
rpart(formula = priceDir ~ snp_cat3 + snp_cat4 + snp_cat5 + bnd_cat4 +
    bnd_cat5 + oil_cat1 + oil_cat2 + oil_cat3 + oil_cat4 + oil_cat5,
    data = trng.d, cp = 0)

Variables actually used in tree construction:
[1] oil_cat1 oil_cat2 oil_cat3 oil_cat4 oil_cat5

Root node error: 1970/4267 = 0.46168

n= 4267

          CP nsplit rel error xerror     xstd
1 0.00126904      0   1.00000 1.0000 0.016531
2 0.00101523      3   0.99594 1.0066 0.016538
3 0.00067682      5   0.99391 1.0071 0.016539
4 0.00000000      8   0.99188 1.0056 0.016537
> table(prediction.rpt, test.d$priceDir)

prediction.rpt High  Low
          High   19   23
          Low  1204 1329
> |
```

Recursive partition



Recursive tree

Our r part tree model is also performing pretty average.

<u>Cross validation</u>

```
> #cross validation
> control <- trainControl(method = "cv", number = 10
> price.rpt.cv <- train(priceDir ~ snp_cat3+ snp_cat4
ata = trng.d , method = "rpart", trControl = control)
> predict.rpt.cv <- predict(price.rpt.cv, newdata = te
> confusionMatrix(predict.rpt.cv, test.d$priceDir)
Confusion Matrix and Statistics

          Reference
Prediction High  Low
      High   20   26
      Low  1203 1326

               Accuracy : 0.5227
                 95% CI : (0.5032, 0.5422)
    No Information Rate : 0.525
    P-Value [Acc > NIR] : 0.6013

                  Kappa : -0.003
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.016353
            Specificity : 0.980769
         Pos Pred Value : 0.434783
         Neg Pred Value : 0.524318
             Prevalence : 0.474951
         Detection Rate : 0.007767
   Detection Prevalence : 0.017864
      Balanced Accuracy : 0.498561

       'Positive' Class : High
```

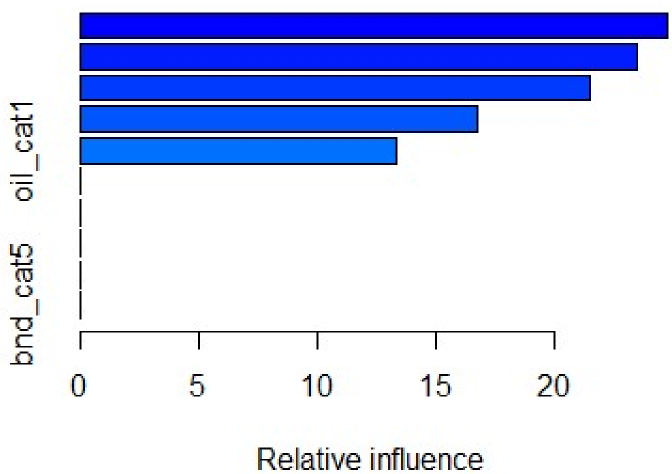
Cross validation

Cross validating our model 10 folds increases the specificity to 0.98.

Gradient boosting model

```
Console F:/Study/SDM/HW4/
> # Modelling using Gradient Boosting
> #install.packages('gbm')
> library('gbm')
> model.gbm <- gbm((unclass(priceDir)-1) ~ sn
cat2 + oil_cat3 + oil_cat4 + oil_cat5, data=t
Distribution not specified, assuming bernoull
> prediction.gbm <- predict(model.gbm, newdat
> head(prediction.gbm[])
[1] 0.5370144 0.5370144 0.5370144 0.5370144 0
> tail(prediction.gbm[])
[1] 0.5370144 0.5370144 0.5370144 0.5370144 0
> summary(model.gbm,n.trees=5000)
              var  rel.inf
oil_cat5 oil_cat5 24.82276
oil_cat4 oil_cat4 23.55841
oil_cat3 oil_cat3 21.55847
oil_cat2 oil_cat2 16.74305
oil_cat1 oil_cat1 13.31731
snp_cat3 snp_cat3  0.00000
snp_cat4 snp_cat4  0.00000
snp_cat5 snp_cat5  0.00000
bnd_cat4 bnd_cat4  0.00000
bnd_cat5 bnd_cat5  0.00000
> |
```

Gradient boosting model



summary of gradient boosting model