IEOR 242 - ICO success prediction project
Annotated code

December 22, 2019

0.1 Annotated R Code

0.1.1 Dead coin prediction

In [2]:
```
library(caret)
library(MASS)
library(caTools)
library(randomForest)
library(ggplot2)
library(GGally)
library(car)
library(rpart)
library(rattle)
library(boot)
library(dplyr)
library(ROCR)

mean_squared_error <- function(responses, predictions) {
  MSE <- mean(((responses - predictions))^2)
  return(MSE)
}

mean_absolute_error <- function(responses, predictions) {
  MAE <- mean(abs(responses - predictions))
  return(MAE)
}

OS_R_squared <- function(responses, predictions, train_responses) {
  baseline <- mean(train_responses)
  SSE <- sum((responses - predictions)^2)
  SST <- sum((responses - baseline)^2)
  r2 <- 1 - SSE/SST
  return(r2)
}

all_metrics <- function(responses, predictions, train_responses) {
  filter_vec = !is.na(responses) & !is.na(predictions)
```

```
  responses <- responses[filter_vec]
  predictions <- predictions[filter_vec]
  train_responses <- train_responses[filter_vec]
  mse <- mean_squared_error(responses, predictions)
  mae <- mean_absolute_error(responses, predictions)
  OSR2 <- OS_R_squared(responses, predictions, train_responses)
  return(c(mse, mae, OSR2))
}

tableAccuracy <- function(label, pred) {
  t = table(label, pred)
  a = sum(diag(t))/length(label)
  return(a)
}

tableTPR <- function(label, pred) {
  t = table(label, pred)
  return(t[2,2]/(t[2,1] + t[2,2]))
}

tableFPR <- function(label, pred) {
  t = table(label, pred)
  return(t[1,2]/(t[1,1] + t[1,2]))
}
```

We load the preprocessed data:

```
In [3]: df <- read.csv("../python/coincheckup/dataset.csv", stringsAsFactors=FALSE, na="")
        df$label_disappeared = as.factor(df$label_disappeared)
        # name and Symbol are the only strings
```

a) Logistic regression

We begin with a logistic regression on the features which are numeric and nhave a very low rate of NAs:

```
In [4]: set.seed(42)
        split = sample.split(df$label_disappeared, SplitRatio = 0.7)
        train_data <- filter(df, split== TRUE)
        test_data <- filter(df, split== FALSE)

        table(train_data$label_disappeared)
        table(test_data$label_disappeared)

        #LOGISTIC
        logistic <- glm(label_disappeared ~ Price + X1h + X24h + X7d + X14d + X30d + X45d +
                        X90d + X200d + Mkt..Cap + X24h.Vol + Circ..Supply + Total.Supply +
                        Team + Product + Coin + Social + Communication + Business +
                        Avg..volume + Age..mo., data = train_data, family="binomial")
```

```r
summary(logistic)

pred = predict(logistic, newdata = test_data, type="response")

table(test_data$label_disappeared, pred>0.2)

tableAccuracy(test_data$label_disappeared, pred>0.2)
tableTPR(test_data$label_disappeared, pred>0.2)
tableFPR(test_data$label_disappeared, pred>0.2)

vif(logistic)
```

```
FALSE  TRUE
 1025   208



FALSE  TRUE
  439    89


Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”


Call:
glm(formula = label_disappeared ~ Price + X1h + X24h + X7d +
    X14d + X30d + X45d + X90d + X200d + Mkt..Cap + X24h.Vol +
    Circ..Supply + Total.Supply + Team + Product + Coin + Social +
    Communication + Business + Avg..volume + Age..mo., family = "binomial",
    data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4129  -0.6457  -0.4438  -0.2806   2.6774

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    4.904e-02  4.592e-01   0.107  0.91494
Price         -5.873e-03  1.089e-02  -0.539  0.58957
X1h            7.936e-05  5.658e-03   0.014  0.98881
X24h           4.684e-04  2.211e-03   0.212  0.83221
X7d            2.764e-03  1.811e-03   1.526  0.12690
X14d          -5.093e-04  1.361e-03  -0.374  0.70821
X30d           5.891e-04  2.114e-03   0.279  0.78049
X45d          -6.051e-03  2.321e-03  -2.608  0.00912 **
X90d           2.879e-03  1.871e-03   1.539  0.12384
X200d         -2.364e-03  2.256e-03  -1.048  0.29464
```

```
Mkt..Cap      -5.270e-05  3.726e-04  -0.141  0.88753
X24h.Vol      -4.836e-04  4.349e-04  -1.112  0.26623
Circ..Supply   5.619e-05  4.204e-04   0.134  0.89366
Total.Supply   2.683e-14  4.072e-14   0.659  0.51002
Team          -2.397e-02  1.576e-02  -1.520  0.12847
Product       -4.131e-03  4.869e-03  -0.848  0.39625
Coin           7.949e-03  5.937e-03   1.339  0.18066
Social        -1.858e-02  8.662e-03  -2.145  0.03199 *
Communication -3.273e-03  3.600e-03  -0.909  0.36335
Business      -3.909e-04  3.243e-03  -0.121  0.90407
Avg..volume   -8.158e-11  8.827e-10  -0.092  0.92636
Age..mo.       7.262e-03  5.277e-03   1.376  0.16878
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1087.97  on 1191  degrees of freedom
Residual deviance:  954.15  on 1170  degrees of freedom
  (41 observations deleted due to missingness)
AIC: 998.15

Number of Fisher Scoring iterations: 11
```

```
        FALSE TRUE
  FALSE   307  117
  TRUE     31   55
```

0.685606060606061

0.63953488372093

0.275943396226415

Price 1.02584781560209 **X1h** 1.15769836383294 **X24h** 2.96033449373743 **X7d** 2.95185035557887 **X14d** 3.24090275717362 **X30d** 4.91817668782566 **X45d** 4.59783289757329 **X90d** 2.61198284261179 **X200d** 1.91331591388318 **Mkt..Cap** 1.03647241353168 **X24h.Vol** 1.03491142934378 **Circ..Supply** 1.05018223690908 **Total.Supply** 1.03860840602233 **Team** 7.10648538827917 **Product** 1.81658646390026 **Coin** 2.19831931567567 **Social** 3.19603310898613 **Communication** 1.46875216267816 **Business** 1.5922862721872 **Avg..volume** 1.0164595410715 **Age..mo.** 1.3693268345375

Then we reduce the range of features to the relevant ones and make a final logistic model, that we compare to the baseline:

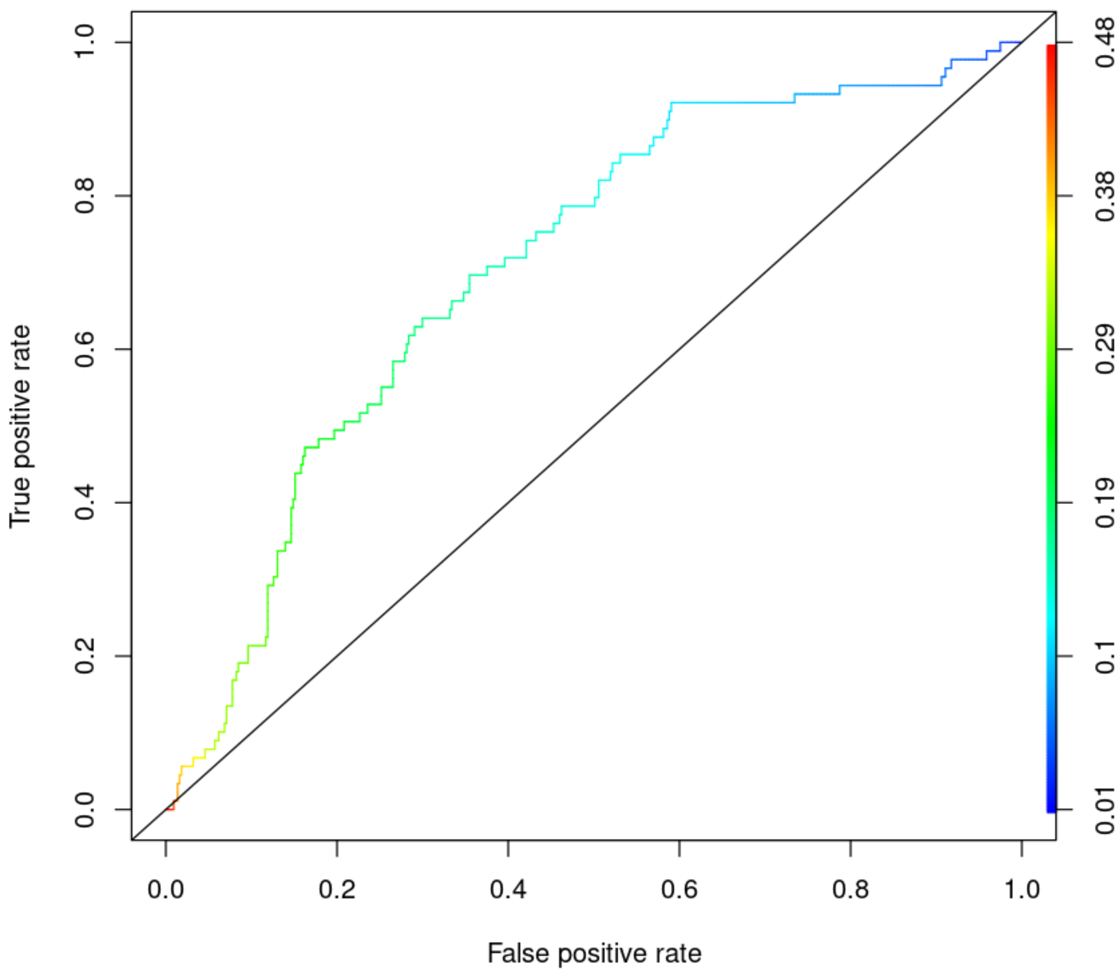
```
In [5]: logistic <- glm(label_disappeared ~ X45d + Social + 0, data = train_data,
                   family="binomial")
        summary(logistic)
```

```r
pred = predict(logistic, newdata = test_data, type="response")

table(test_data$label_disappeared, pred>0.2)

tableAccuracy(test_data$label_disappeared, pred>0.2)
tableTPR(test_data$label_disappeared, pred>0.2)
tableFPR(test_data$label_disappeared, pred>0.2)

# Baseline accuracy:
t_ <- table(test_data$label_disappeared)
t_[1]/sum(t_)

rocr.pred <- prediction(pred, test_data$label_disappeared)
perf <- performance(rocr.pred, "tpr", "fpr")
plot(perf, colorize = TRUE)
abline(0, 1)
as.numeric(performance(rocr.pred, "auc")@y.values)
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”


Call:
glm(formula = label_disappeared ~ X45d + Social + 0, family = "binomial",
    data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1773  -0.6347  -0.4854  -0.3496   2.7011

Coefficients:
        Estimate Std. Error z value Pr(>|z|)
X45d   -0.004062   0.001108  -3.667 0.000245 ***
Social -0.032039   0.001607 -19.935  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1696.8  on 1224  degrees of freedom
Residual deviance: 1011.9  on 1222  degrees of freedom
  (9 observations deleted due to missingness)
AIC: 1015.9

Number of Fisher Scoring iterations: 8
```

```
      FALSE TRUE
FALSE   320  117
TRUE     37   52
```

0.704545454545455
0.584269662921348
0.267734553775744
FALSE: 0.831439393939394
0.708919342812334



b) Random Forest

Let's begin checking the number of NAs in each column to select the features.

```
In [10]: sapply(df, function(x){ sum(is.na(x)) })
```

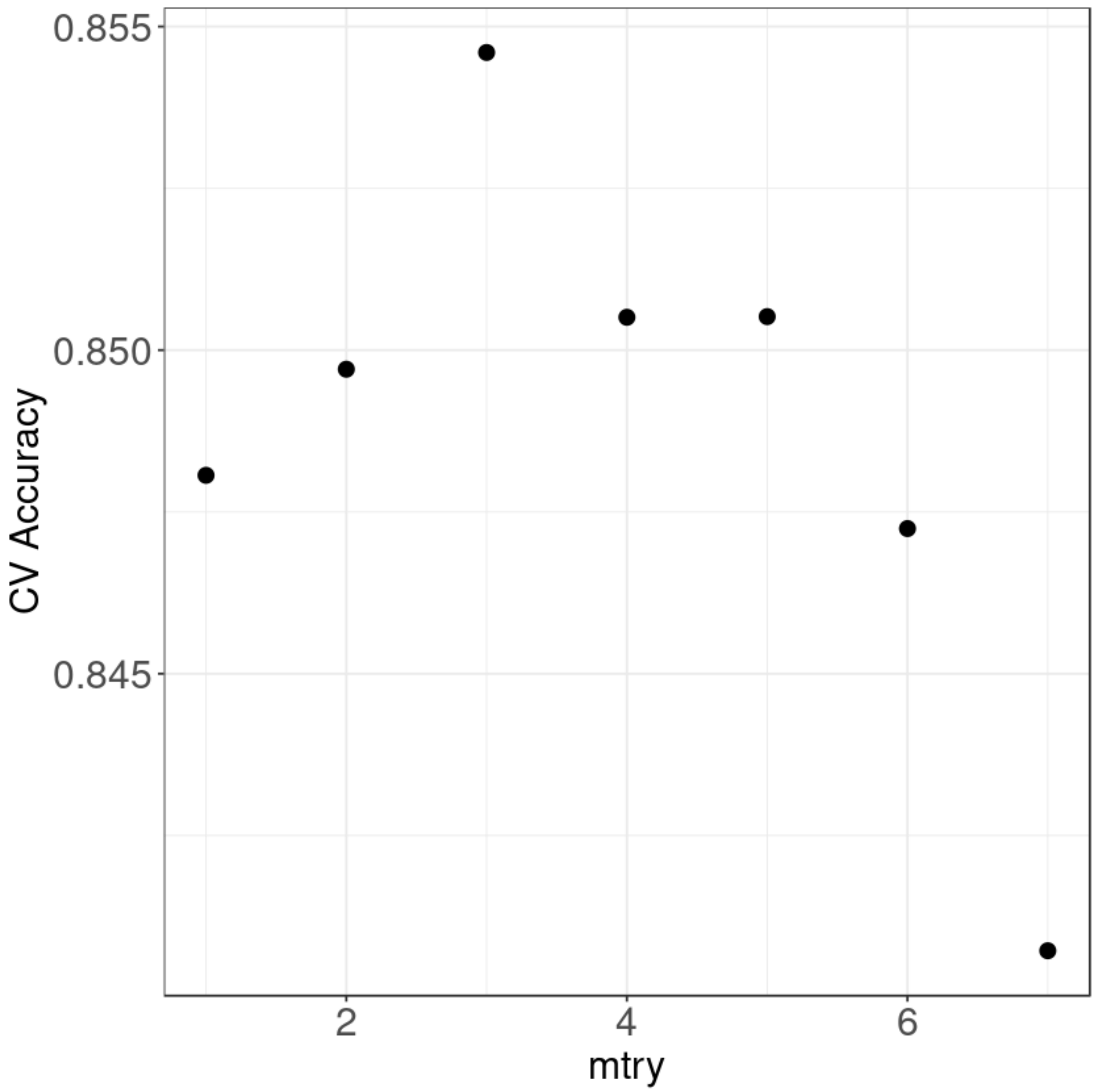
Name 0 **MC..** 0 **Symbol** 0 **Price** 0 **BTC** 0 **X1h** 0 **X24h** 0 **X7d** 0 **X14d** 0 **X30d** 4 **X45d** 11 **X90d** 22 **X200d** 48 **Mkt..Cap** 0 **MCAP.BTC** 0 **X24h.Vol** 0 **X24h.Vol.BTC** 0 **Circ..Supply** 0 **Total.Supply** 9 **Max..Supply** 1394 **Team** 0 **Advisors** 566 **Brand.Buzz** 566 **Product** 0 **Coin** 0 **Social** 0 **Communication** 0 **Business** 0 **GitHub** 697 **GitHub.1** 697 **Avg..volume** 0 **Age..mo.** 1 **Winning.months** 1 **label_Price** 297 **label_Mkt..Cap** 297 **label_growth_rate_Price** 297 **label_growth_rate_Mkt..Cap** 297 **label_disappeared** 0

Now let's try a random forest with cross-validation. The dataset is not balanced so we try a cross-validation measured with Accuracy and then F1-score as Accuracy is less relevant in this case.

```
In [11]: set.seed(42)
         test_data_filled_with_0 <- test_data
         test_data_filled_with_0[is.na(test_data_filled_with_0)] <- 0
         train_data.mm = as.data.frame(model.matrix(label_disappeared ~ X24h + X14d + X45d +
                 Social + Mkt..Cap + Age..mo. + Business, data = train_data))
         test_data.mm = as.data.frame(model.matrix(label_disappeared ~ X24h + X14d + X45d +
                 Social + Mkt..Cap + Age..mo. + Business, data = test_data_filled_with_0))

         train.rf <- train(label_disappeared ~ X24h + X14d + X45d + Social + Mkt..Cap +
                           Age..mo. + Business,
                           data = train_data,
                           method = "rf",
                           na.action  = na.omit,
                           tuneGrid = data.frame(mtry=1:7),
                           trControl = trainControl(method="cv", number=5),
                           metric = "Accuracy")
         train.rf$results
         train.rf
         best.rf <- train.rf$finalModel
         pred.rf <- predict(best.rf, newdata = test_data.mm) # can use same model matrix

         ggplot(train.rf$results, aes(x = mtry, y = Accuracy)) + geom_point(size = 3) +
           ylab("CV Accuracy") + theme_bw() + theme(axis.title=element_text(size=18),
                           axis.text=element_text(size=18))
```

mtry <int>	Accuracy <dbl>	Kappa <dbl>	AccuracySD <dbl>	KappaSD <dbl>
1	0.8480651	0.1820981	0.01230487	0.07219353
2	0.8497045	0.2675792	0.01730529	0.08837830
3	0.8545991	0.3131013	0.01457905	0.07484737
4	0.8505074	0.2927120	0.01216925	0.05062367
5	0.8505175	0.3181606	0.01943042	0.07520890
6	0.8472421	0.2995333	0.01863631	0.06998712
7	0.8407181	0.2733978	0.02110235	0.08098173

```
Random Forest

1233 samples
   7 predictor
   2 classes: 'FALSE', 'TRUE'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 980, 978, 980, 979, 979
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  1     0.8480651  0.1820981
  2     0.8497045  0.2675792
  3     0.8545991  0.3131013
  4     0.8505074  0.2927120
  5     0.8505175  0.3181606
  6     0.8472421  0.2995333
  7     0.8407181  0.2733978

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 3.
```

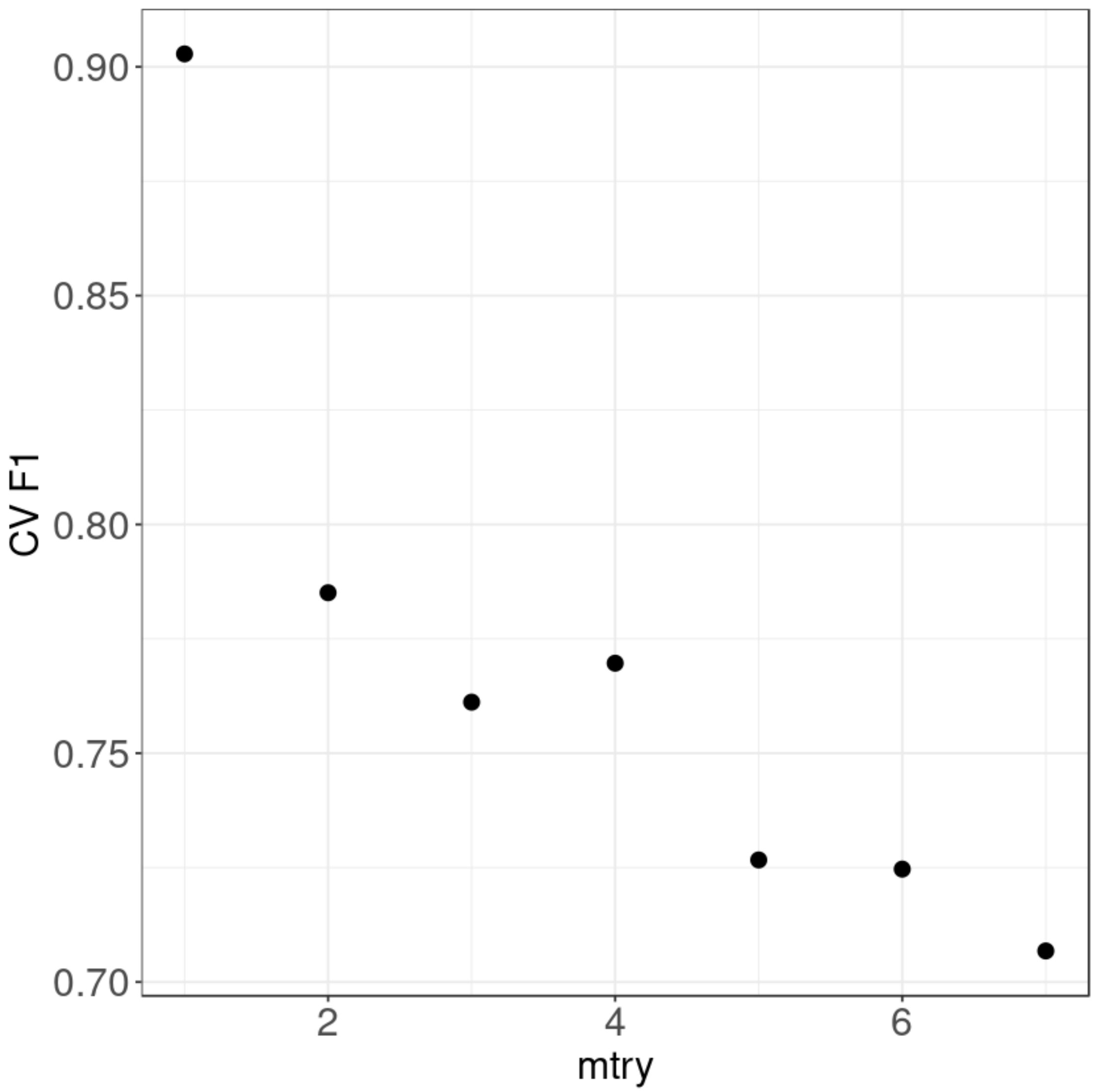



```
In [14]: f1 <- function (data, lev = NULL, model = NULL) {
           pred <- data$pred[!is.na(data$pred)&!is.na(data$obs)]
           obs <- data$obs[!is.na(data$pred)&!is.na(data$obs)]
           precision <- posPredValue(pred, obs, positive = lev[2])
           recall  <- sensitivity(pred, obs, postive = lev[1])
           f1_val <- (2 * precision * recall) / (precision + recall)
           names(f1_val) <- c("F1")
           #print(precision)
           #print(recall)
           #print(f1_val)
           f1_val
         }
```

```
train.rf <- train(label_disappeared ~ X24h + X14d + X45d + Social + Mkt..Cap +
                  Age..mo. + Business,
                  data = train_data,
                  method = "rf",
                  na.action  = na.omit,
                  tuneGrid = data.frame(mtry=1:7),
                  trControl = trainControl(method="cv", number=5, summaryFunction = f1)
                  metric = PYl+s"F1")
train.rf$results
train.rf
best.rf <- train.rf$finalModel
pred.rf <- predict(best.rf, newdata = test_data.mm) # can use same model matrix

ggplot(train.rf$results, aes(x = mtry, y = F1)) + geom_point(size = 3) +
  ylab("CV F1") + theme_bw() + theme(axis.title=element_text(size=18),
  axis.text=element_text(size=18))
```

mtry <int>	F1 <dbl>	F1SD <dbl>
1	0.9028305	0.08709772
2	0.7850726	0.07805387
3	0.7611599	0.11121317
4	0.7696718	0.06996851
5	0.7266709	0.10382716
6	0.7246719	0.11886381
7	0.7067876	0.12034424

```
Random Forest

1233 samples
   7 predictor
   2 classes: 'FALSE', 'TRUE'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 979, 980, 979, 978, 980
Resampling results across tuning parameters:

  mtry  F1
  1     0.9028305
  2     0.7850726
  3     0.7611599
  4     0.7696718
  5     0.7266709
  6     0.7246719
  7     0.7067876

F1 was used to select the optimal model using the largest value.
The final value used for the model was mtry = 1.
```



An mtry of 3 looks good. We can try sampling stratification to handle the imbalance of the data.

```
In [15]: set.seed(42)
         rare.class.prevalence = 0.2
         nRareSamples = 1000 * rare.class.prevalence
         mod.rf <- randomForest(label_disappeared ~ X24h + X14d + X45d + Social + Mkt..Cap +
                 Age..mo. + Business, data = train_data, mtry = 3, nodesize = 5, ntree = 1000,
                 strata=train_data$label_disappeared,
                 sampsize=c(nRareSamples,nRareSamples), na.action = na.omit)
         # print(mod.rf)
```

```
pred.rf <- predict(mod.rf, newdata = test_data)

table(test_data$label_disappeared, pred.rf)

tableAccuracy(test_data$label_disappeared, pred.rf)
tableTPR(test_data$label_disappeared, pred.rf)
tableFPR(test_data$label_disappeared, pred.rf)
```

```
       pred.rf
        FALSE TRUE
  FALSE   427   10
  TRUE     70   19
```

0.84469696969697
0.213483146067416
0.022883295194508

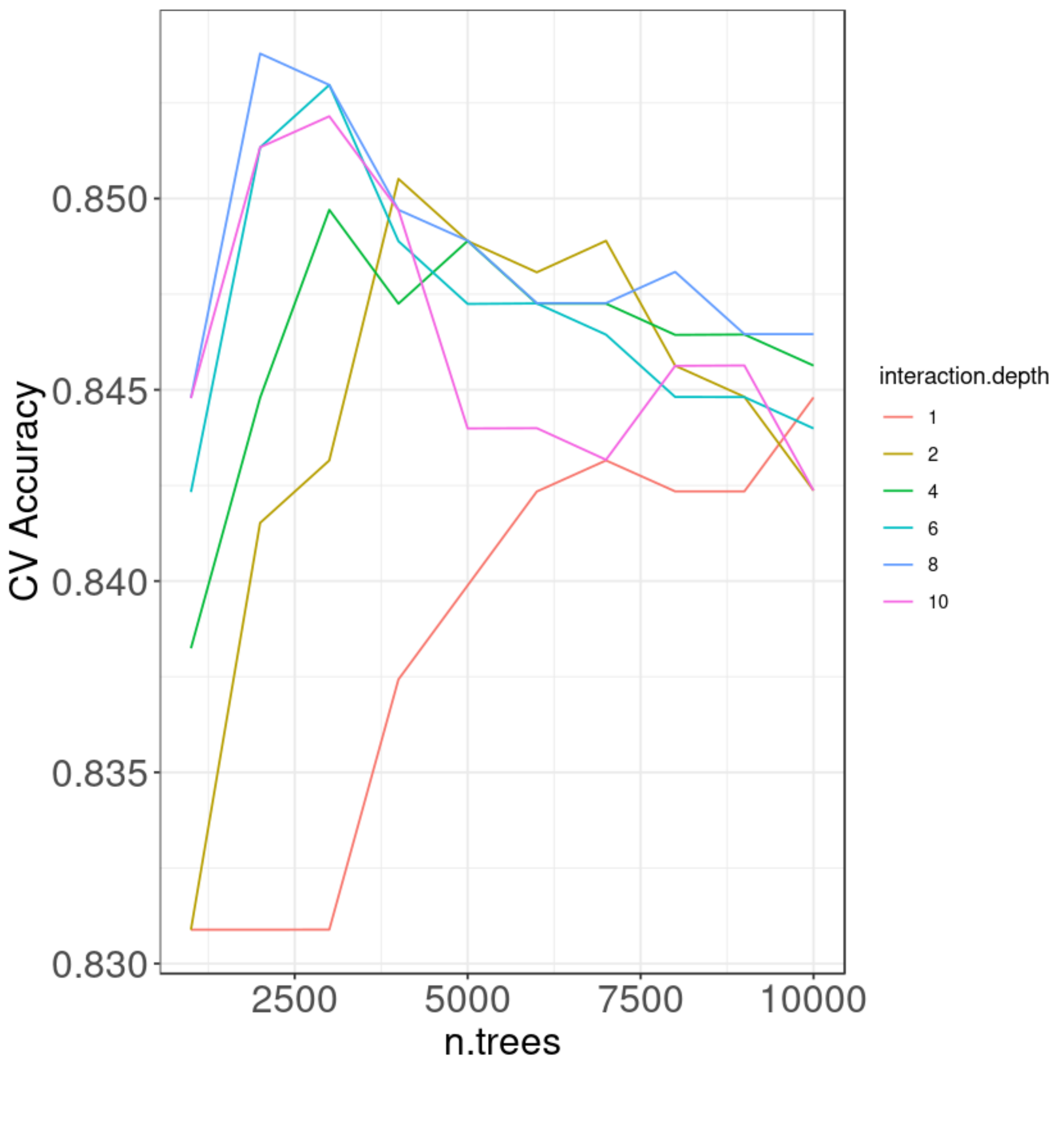
c) Boosting

Finally we can try a boosting method with cross-validation.

```
In [17]: tGrid = expand.grid(n.trees = (1:10)*1000, interaction.depth = c(1,2,4,6,8,10),
                             shrinkage = 0.001, n.minobsinnode = 10)

         set.seed(42)
         train.boost <- train(label_disappeared ~ X24h + X14d + X45d + Social + Mkt..Cap +
                              Age..mo. + Business,
                              data = train_data,
                              method = "gbm",    ## gradient boosting machine
                              tuneGrid = tGrid,
                              trControl = trainControl(method="cv", number=5),
                              metric = "Accuracy",
                              na.action = na.omit)
         train.boost
         best.boost <- train.boost$finalModel

         ggplot(train.boost$results, aes(x = n.trees, y = Accuracy, colour =
           as.factor(interaction.depth))) + geom_line() +
           ylab("CV Accuracy") + theme_bw() + theme(axis.title=element_text(size=18),
           axis.text=element_text(size=18)) +
           scale_color_discrete(name = "interaction.depth")
```

```
Stochastic Gradient Boosting

1233 samples
   7 predictor
   2 classes: 'FALSE', 'TRUE'
```

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 980, 978, 980, 979, 979

Tuning parameter 'shrinkage' was held constant at a value of 0.001

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were n.trees = 2000, interaction.depth =
 8, shrinkage = 0.001 and n.minobsinnode = 10.
```



We choose n.trees = 2000, interaction.depth = 8.

```
In [18]: pred.best.boost <- predict(best.boost, newdata = test_data.mm, n.trees = 2000,
```

```
interaction.depth = 8) # from CV plot

table(test_data$label_disappeared, pred.best.boost>0.7)

tableAccuracy(test_data$label_disappeared, pred.best.boost>0.7)
tableTPR(test_data$label_disappeared, pred.best.boost>0.7)
tableFPR(test_data$label_disappeared, pred.best.boost>0.7)
```

```
       FALSE TRUE
  FALSE   221  218
  TRUE     70   19
```

0.454545454545455
0.213483146067416
0.496583143507973

0.1.2 Portfolio determination based on naive growth rates

a) Regression

We try a linear model on a lot of features.

```
In [22]: set.seed(42)
         split = sample.split(df$label_growth_rate_Price, SplitRatio = 0.7)
         price_train_data <- filter(df, split== TRUE)
         price_test_data <- filter(df, split== FALSE)

         price_train_data = price_train_data %>% filter(as.integer(label_disappeared)==1)
                 # we keep label_disappeared = FALSE
         price_test_data = price_test_data %>% filter(as.integer(label_disappeared)==1)

         mod <- lm(label_growth_rate_Price ~ Price + X1h + X24h + X7d + X14d + X30d + X45d +
                         X90d + X200d + Mkt..Cap + X24h.Vol + Circ..Supply + Total.Supply +
                         Team + Product + Coin + Social + Communication + Business +
                         Avg..volume + Age..mo., data = price_train_data)

         summary(mod)

         pred_lm = predict(mod, newdata = price_test_data)

         all_metrics(price_test_data$label_growth_rate_Price, pred_lm,
                         price_train_data$label_growth_rate_Price)
```

```
Call:
lm(formula = label_growth_rate_Price ~ Price + X1h + X24h + X7d +
    X14d + X30d + X45d + X90d + X200d + Mkt..Cap + X24h.Vol +
```

```
    Circ..Supply + Total.Supply + Team + Product + Coin + Social +
    Communication + Business + Avg..volume + Age..mo., data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-95.323  -0.448   0.006   0.504 157.521

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.380e+00  1.463e+00  -0.943    0.346
Price         -1.363e-05  9.791e-04  -0.014    0.989
X1h            2.563e-02  2.873e-02   0.892    0.373
X24h          -1.096e-02  9.764e-03  -1.122    0.262
X7d           -3.442e-03  7.134e-03  -0.482    0.630
X14d           9.311e-04  3.500e-03   0.266    0.790
X30d           5.279e-04  4.867e-03   0.108    0.914
X45d          -2.804e-03  2.983e-03  -0.940    0.347
X90d           5.591e-03  5.494e-03   1.018    0.309
X200d         -2.616e-03  5.139e-03  -0.509    0.611
Mkt..Cap       4.964e-05  1.062e-03   0.047    0.963
X24h.Vol       9.059e-06  1.117e-03   0.008    0.994
Circ..Supply  -1.920e-03  1.103e-03  -1.741    0.082 .
Total.Supply   9.642e-12  3.189e-13  30.230   <2e-16 ***
Team           1.977e-02  4.204e-02   0.470    0.638
Product       -6.773e-03  1.328e-02  -0.510    0.610
Coin          -8.403e-03  1.685e-02  -0.499    0.618
Social         3.590e-03  2.370e-02   0.151    0.880
Communication -5.365e-03  8.557e-03  -0.627    0.531
Business       6.493e-03  1.123e-02   0.578    0.563
Avg..volume    2.020e-11  1.844e-09   0.011    0.991
Age..mo.       1.896e-02  1.633e-02   1.161    0.246
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.24 on 962 degrees of freedom
  (40 observations deleted due to missingness)
Multiple R-squared:  0.4896,Adjusted R-squared:  0.4785
F-statistic: 43.95 on 21 and 962 DF,  p-value: < 2.2e-16
```

1. 582.295129943916 2. 2.15332619393653 3. -173.530306079791

```
In [23]: vif(mod)
```

Price 1.02423943344359 **X1h** 1.27695002473311 **X24h** 2.20857264825821 **X7d** 2.20262213596122 **X14d** 51.0399053125082 **X30d** 57.8341308908584 **X45d** 16.3502969281802 **X90d** 6.77458705855804 **X200d** 3.77206853766809 **Mkt..Cap** 1.02694209743189 **X24h.Vol** 1.02876014673704 **Circ..Supply**

1.07660848685341 **Total.Supply** 1.01072167422902 **Team** 6.49637787342385 **Product** 1.9710382752697 **Coin** 1.90590139973695 **Social** 3.14333047583439 **Communication** 1.53408207175482 **Business** 1.36185221776999 **Avg..volume** 1.02165670821974 **Age..mo.** 1.3028447664378

This is not good, there is some multicolinearity. Let's remove the guilty features.

```
In [26]: mod <- lm(label_growth_rate_Price ~ Price + X1h + X24h + X7d + X30d + X90d + X200d +
                      Mkt..Cap + X24h.Vol + Circ..Supply + Total.Supply + Team + Product +
                      Coin + Social + Communication + Business + Avg..volume + Age..mo.,
                      data = price_train_data)

         summary(mod)

         pred_lm = predict(mod, newdata = price_test_data)

         all_metrics(price_test_data$label_growth_rate_Price, pred_lm,
                         price_train_data$label_growth_rate_Price)
```

```
Call:
lm(formula = label_growth_rate_Price ~ Price + X1h + X24h + X7d +
    X30d + X90d + X200d + Mkt..Cap + X24h.Vol + Circ..Supply +
    Total.Supply + Team + Product + Coin + Social + Communication +
    Business + Avg..volume + Age..mo., data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-95.222  -0.426  -0.009   0.496 157.685

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.506e+00  1.452e+00  -1.037   0.2999
Price         -1.311e-05  9.783e-04  -0.013   0.9893
X1h            2.299e-02  2.857e-02   0.805   0.4212
X24h          -9.303e-03  9.536e-03  -0.976   0.3295
X7d           -3.687e-03  7.124e-03  -0.517   0.6049
X30d          -8.050e-05  9.549e-04  -0.084   0.9328
X90d           1.554e-03  3.466e-03   0.448   0.6541
X200d         -4.926e-04  4.608e-03  -0.107   0.9149
Mkt..Cap       5.801e-05  1.061e-03   0.055   0.9564
X24h.Vol       7.150e-05  1.114e-03   0.064   0.9489
Circ..Supply  -1.902e-03  1.102e-03  -1.726   0.0846 .
Total.Supply   9.631e-12  3.186e-13  30.233   <2e-16 ***
Team           1.702e-02  4.190e-02   0.406   0.6847
Product       -6.461e-03  1.327e-02  -0.487   0.6263
Coin          -7.157e-03  1.676e-02  -0.427   0.6695
Social         5.232e-03  2.359e-02   0.222   0.8245
Communication -5.266e-03  8.549e-03  -0.616   0.5381
```

```
Business       6.734e-03  1.122e-02   0.600   0.5484
Avg..volume    1.029e-11  1.843e-09   0.006   0.9955
Age..mo.       2.052e-02  1.613e-02   1.272   0.2037
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.236 on 964 degrees of freedom
  (40 observations deleted due to missingness)
Multiple R-squared:  0.4891,Adjusted R-squared:  0.4791
F-statistic: 48.58 on 19 and 964 DF,  p-value: < 2.2e-16
```

1. 564.062942832866 2. 1.90025594030086 3. -168.065604361777

In [27]:
```
vif(mod)
```

Price 1.02383458947674 **X1h** 1.26378621144891 **X24h** 2.10915904620629 **X7d** 2.1991343191775 **X30d** 2.22872902583279 **X90d** 2.69968306031107 **X200d** 3.03647959935625 **Mkt..Cap** 1.02665757634139 **X24h.Vol** 1.02508903973853 **Circ..Supply** 1.07610953486792 **Total.Supply** 1.00943241102529 **Team** 6.46108652227192 **Product** 1.96792266463445 **Coin** 1.88882024934975 **Social** 3.11853701685625 **Communication** 1.53321255914753 **Business** 1.36088711654353 **Avg..volume** 1.02158918501001 **Age..mo.** 1.27210148432654

This still does not look good. Let's narrow down the features.

In [20]:
```
mod <- lm(label_growth_rate_Price ~ Circ..Supply + Total.Supply + 0,
          data = price_train_data)

summary(mod)

pred_lm = predict(mod, newdata = price_test_data)

all_metrics(price_test_data$label_growth_rate_Price, pred_lm,
price_train_data$label_growth_rate_Price)
```

```
Call:
lm(formula = label_growth_rate_Price ~ Circ..Supply + Total.Supply +
    0, data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-96.415  -0.701  -0.407   0.164 158.564

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
Circ..Supply -2.534e-03  8.533e-04   -2.97  0.00305 **
Total.Supply  9.579e-12  3.097e-13   30.93  < 2e-16 ***
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.074 on 1017 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.485,Adjusted R-squared:  0.484
F-statistic: 478.9 on 2 and 1017 DF,  p-value: < 2.2e-16
```

1. 547.59691212069 2. 1.94101815834009 3. -166.769645220578

We can compare it to the baseline (predicting negative all the time). We assume that we invest the same amount of money in each coin, ignoring fees, and compute the result growth rate of our portfolio following our regression strategy and the baseline.

```
In [29]: filter_vec = !is.na(pred_lm)
         sum((pred_lm*price_test_data$label_growth_rate_Price)[filter_vec])
                         /sum(pred_lm[filter_vec])

         # Baseline
         sum(price_test_data$label_growth_rate_Price[filter_vec])
                         /length(price_test_data$label_growth_rate_Price[filter_vec])
```

0.142130885876864

-0.342125271527515

We beat it!

0.1.3 Portfolio determination in a log normal stock model

a) Regression

With our new log-normal assumption we can try to predict the difference of the logarithms of the price between March and December.

```
In [30]: df$price_log_diff = log(df$label_Price) - log(df$Price) # follow mu T + sigma B_T

         split = sample.split(df$price_log_diff, SplitRatio = 0.7)
         price_train_data <- filter(df, split== TRUE)
         price_test_data <- filter(df, split== FALSE)

         price_train_data = price_train_data %>% filter(as.integer(label_disappeared)==1)
         # we keep label_disappeared = FALSE
         price_test_data = price_test_data %>% filter(as.integer(label_disappeared)==1)

         mod_log <- lm(price_log_diff ~ Price + X1h + X24h + X7d + X14d + X30d + X45d + X90d +
                 X200d + Mkt..Cap + X24h.Vol + Circ..Supply + Total.Supply + Team +
                 Product + Coin + Social + Communication + Business + Avg..volume + Age..mo.,
                 data = price_train_data)

         summary(mod_log)
```

```
pred_lm = predict(mod_log, newdata = price_test_data)

all_metrics(price_test_data$price_log_diff, pred_lm, price_train_data$price_log_diff)
```

```
Call:
lm(formula = price_log_diff ~ Price + X1h + X24h + X7d + X14d +
    X30d + X45d + X90d + X200d + Mkt..Cap + X24h.Vol + Circ..Supply +
    Total.Supply + Team + Product + Coin + Social + Communication +
    Business + Avg..volume + Age..mo., data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-6.2257 -0.4446  0.0628  0.5709  3.8110

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -9.089e-01  1.915e-01  -4.747 2.37e-06 ***
Price          2.427e-05  3.484e-05   0.697   0.4861
X1h           -3.607e-03  5.843e-03  -0.617   0.5372
X24h          -3.514e-03  1.624e-03  -2.164   0.0307 *
X7d           -1.298e-03  1.028e-03  -1.262   0.2074
X14d          -6.971e-04  3.174e-04  -2.196   0.0283 *
X30d           5.989e-04  4.839e-04   1.238   0.2161
X45d           6.502e-05  2.585e-04   0.252   0.8014
X90d           5.886e-04  5.132e-04   1.147   0.2517
X200d         -9.859e-04  5.229e-04  -1.885   0.0597 .
Mkt..Cap      -2.969e-04  1.388e-04  -2.139   0.0327 *
X24h.Vol      -7.968e-07  1.410e-04  -0.006   0.9955
Circ..Supply   2.499e-04  1.455e-04   1.718   0.0862 .
Total.Supply   1.668e-14  1.874e-14   0.890   0.3738
Team           7.936e-03  5.567e-03   1.426   0.1543
Product       -1.076e-03  1.750e-03  -0.615   0.5385
Coin          -2.972e-03  2.195e-03  -1.354   0.1761
Social        -7.665e-04  3.138e-03  -0.244   0.8071
Communication  3.429e-04  1.144e-03   0.300   0.7645
Business      -3.736e-03  1.468e-03  -2.544   0.0111 *
Avg..volume    1.576e-10  1.418e-10   1.112   0.2665
Age..mo.       3.241e-03  2.212e-03   1.465   0.1433
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9755 on 978 degrees of freedom
  (37 observations deleted due to missingness)
Multiple R-squared:  0.05583,Adjusted R-squared:  0.03555
F-statistic: 2.754 on 21 and 978 DF,  p-value: 3.95e-05
```

1. 1.37419565150063 2. 0.786600323768397 3. -0.0510270204478771

```
In [31]: vif(mod_log)
```

Price 1.06407441227154 **X1h** 1.07115978177007 **X24h** 1.22544188241185 **X7d** 1.31330264496053 **X14d** 753.264205918813 **X30d** 1730.32601194076 **X45d** 491.069203199915 **X90d** 10.6815410522711 **X200d** 3.59844428634785 **Mkt..Cap** 1.0247473315568 **X24h.Vol** 1.02517852100103 **Circ..Supply** 1.07871229383061 **Total.Supply** 1.01891666805316 **Team** 6.3719217947138 **Product** 1.92537905891746 **Coin** 1.86682147289798 **Social** 3.14543604788404 **Communication** 1.52506691080304 **Business** 1.3559062673666 **Avg..volume** 1.09292020780236 **Age..mo.** 1.31230648270897

```
In [34]: mod_log <- lm(price_log_diff ~ Price + X1h + X24h + X7d + X14d + X90d + X200d +
                        Mkt..Cap + X24h.Vol + Circ..Supply + Total.Supply + Team + Product +
                        Coin + Social + Communication + Business + Avg..volume + Age..mo.,
                        data = price_train_data)

         summary(mod_log)

         pred_lm = predict(mod_log, newdata = price_test_data)

         all_metrics(price_test_data$price_log_diff, pred_lm, price_train_data$price_log_diff)
```

```
Call:
lm(formula = price_log_diff ~ Price + X1h + X24h + X7d + X14d +
    X90d + X200d + Mkt..Cap + X24h.Vol + Circ..Supply + Total.Supply +
    Team + Product + Coin + Social + Communication + Business +
    Avg..volume + Age..mo., data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-6.1966 -0.4454  0.0553  0.5753  3.7657

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -9.555e-01  1.909e-01  -5.005 6.62e-07 ***
Price          2.525e-05  3.491e-05   0.723 0.469718
X1h           -3.627e-03  5.851e-03  -0.620 0.535459
X24h          -3.597e-03  1.626e-03  -2.212 0.027225 *
X7d           -1.328e-03  1.029e-03  -1.291 0.196887
X14d          -4.881e-05  2.479e-05  -1.969 0.049266 *
X90d           8.115e-04  4.038e-04   2.010 0.044752 *
X200d         -1.469e-03  4.261e-04  -3.448 0.000589 ***
Mkt..Cap      -2.973e-04  1.391e-04  -2.138 0.032777 *
X24h.Vol      -6.626e-06  1.412e-04  -0.047 0.962579
Circ..Supply   2.468e-04  1.458e-04   1.693 0.090731 .
```

```
Total.Supply   1.546e-14  1.874e-14   0.825 0.409431
Team           8.054e-03  5.568e-03   1.446 0.148377
Product       -1.288e-03  1.751e-03  -0.735 0.462238
Coin          -2.875e-03  2.198e-03  -1.308 0.191143
Social        -5.695e-04  3.138e-03  -0.181 0.856039
Communication  4.538e-04  1.145e-03   0.396 0.692061
Business      -3.784e-03  1.471e-03  -2.573 0.010242 *
Avg..volume    1.605e-10  1.420e-10   1.130 0.258656
Age..mo.       3.300e-03  2.204e-03   1.497 0.134725
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9776 on 980 degrees of freedom
  (37 observations deleted due to missingness)
Multiple R-squared:  0.04984,Adjusted R-squared:  0.03142
F-statistic: 2.706 on 19 and 980 DF,  p-value: 0.0001092
```

1. 1.37718128083273 2. 0.789451576546088 3. -0.0533105214163541

```
In [35]: vif(mod_log)
```

Price 1.06389350752336 **X1h** 1.06954219074384 **X24h** 1.22325176284725 **X7d** 1.30815268834064 **X14d** 4.57576438585392 **X90d** 6.58402089382897 **X200d** 2.37953671288416 **Mkt..Cap** 1.02437219200182 **X24h.Vol** 1.02414192975333 **Circ..Supply** 1.07809508758456 **Total.Supply** 1.01415980805924 **Team** 6.34760662656953 **Product** 1.92036647546332 **Coin** 1.86268778843273 **Social** 3.13312064680212 **Communication** 1.52216803443362 **Business** 1.3550512673174 **Avg..volume** 1.09262600492207 **Age..mo.** 1.29729721274557

The VIF is okay now. Let's remove some features.

```
In [38]: mod_log <- lm(price_log_diff ~ X24h + X200d + Mkt..Cap + Business,
                       data = price_train_data)

         summary(mod_log)

         pred_lm_log = predict(mod_log, newdata = price_test_data)

         all_metrics(price_test_data$price_log_diff, pred_lm_log,
                        price_train_data$price_log_diff)
```

```
Call:
lm(formula = price_log_diff ~ X24h + X200d + Mkt..Cap + Business,
    data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-6.2460 -0.4330  0.0580  0.5923  3.9618

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.7949174  0.1167271  -6.810 1.68e-11 ***
X24h        -0.0047998  0.0014715  -3.262  0.00114 **
X200d       -0.0008992  0.0002755  -3.265  0.00113 **
Mkt..Cap    -0.0003394  0.0001378  -2.464  0.01392 *
Business    -0.0027869  0.0012598  -2.212  0.02717 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9794 on 1000 degrees of freedom
  (32 observations deleted due to missingness)
Multiple R-squared:  0.03057,Adjusted R-squared:  0.02669
F-statistic: 7.883 on 4 and 1000 DF,  p-value: 2.956e-06
```

1. 1.35643667936919 2. 0.784982884758303 3. -0.0401929076627052

In [39]:
```
filter_vec = !is.na(pred_lm_log)
sum((exp(pred_lm_log)*price_test_data$label_growth_rate_Price)[filter_vec])
        /sum(exp(pred_lm_log)[filter_vec])

# Baseline
sum(price_test_data$label_growth_rate_Price[filter_vec])
        /length(price_test_data$label_growth_rate_Price[filter_vec])
```

0.461436810190696
0.439390893108447
That's fine but not amazing, let's think a bit and try something else.

In [40]:
```
mod_log <- lm(price_log_diff ~ X7d + Mkt..Cap + Age..mo., data = price_train_data)

summary(mod_log)

pred_lm_log = predict(mod_log, newdata = price_test_data)

all_metrics(price_test_data$price_log_diff, pred_lm_log,
                price_train_data$price_log_diff)
```

```
Call:
lm(formula = price_log_diff ~ X7d + Mkt..Cap + Age..mo., data = price_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-6.3005 -0.4493  0.0632  0.5849  3.9844
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.0571347  0.0505030 -20.932   <2e-16 ***
X7d         -0.0016871  0.0009027  -1.869   0.0619 .
Mkt..Cap    -0.0003295  0.0001397  -2.359   0.0185 *
Age..mo.     0.0041164  0.0019420   2.120   0.0343 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9966 on 1033 degrees of freedom
Multiple R-squared:  0.01299,Adjusted R-squared:  0.01012
F-statistic: 4.531 on 3 and 1033 DF,  p-value: 0.003655
```

1. 1.3321906348193 2. 0.766205419022985 3. 0.0152780706337514

In [41]:
```
filter_vec = !is.na(pred_lm_log)
sum((exp(pred_lm_log)*price_test_data$label_growth_rate_Price)[filter_vec])
        /sum(exp(pred_lm_log)[filter_vec])

# Baseline
sum(price_test_data$label_growth_rate_Price[filter_vec])
        /length(price_test_data$label_growth_rate_Price[filter_vec])
```

0.500360688948723
0.412744498999508
Much better already! We see that filtering out coisn where informations are missing, our portfolio behaves already better! That's because trustworthy coins get all their information filled.

b) Random forest

We take the features from the 2 previous models and we try it out on a random forest.

In [50]:
```
price_test_data_filled_with_0 <- price_test_data
price_test_data_filled_with_0[is.na(price_test_data_filled_with_0)] <- 0
price_test_data_log.mm = as.data.frame(model.matrix(price_log_diff ~ X24h + X7d +
        X200d + Mkt..Cap + Age..mo. + Business, data = price_test_data_filled_with_0))

train.rf <- train(price_log_diff ~ X24h + X7d + X200d + Mkt..Cap + Age..mo. +
                  Business,
                  data = price_train_data,
                  method = "rf",
                  na.action  = na.omit,
                  tuneGrid = data.frame(mtry=1:6),
                  trControl = trainControl(method="cv", number=5),
                  distribution="gaussian",
                  metric = "RMSE")
```

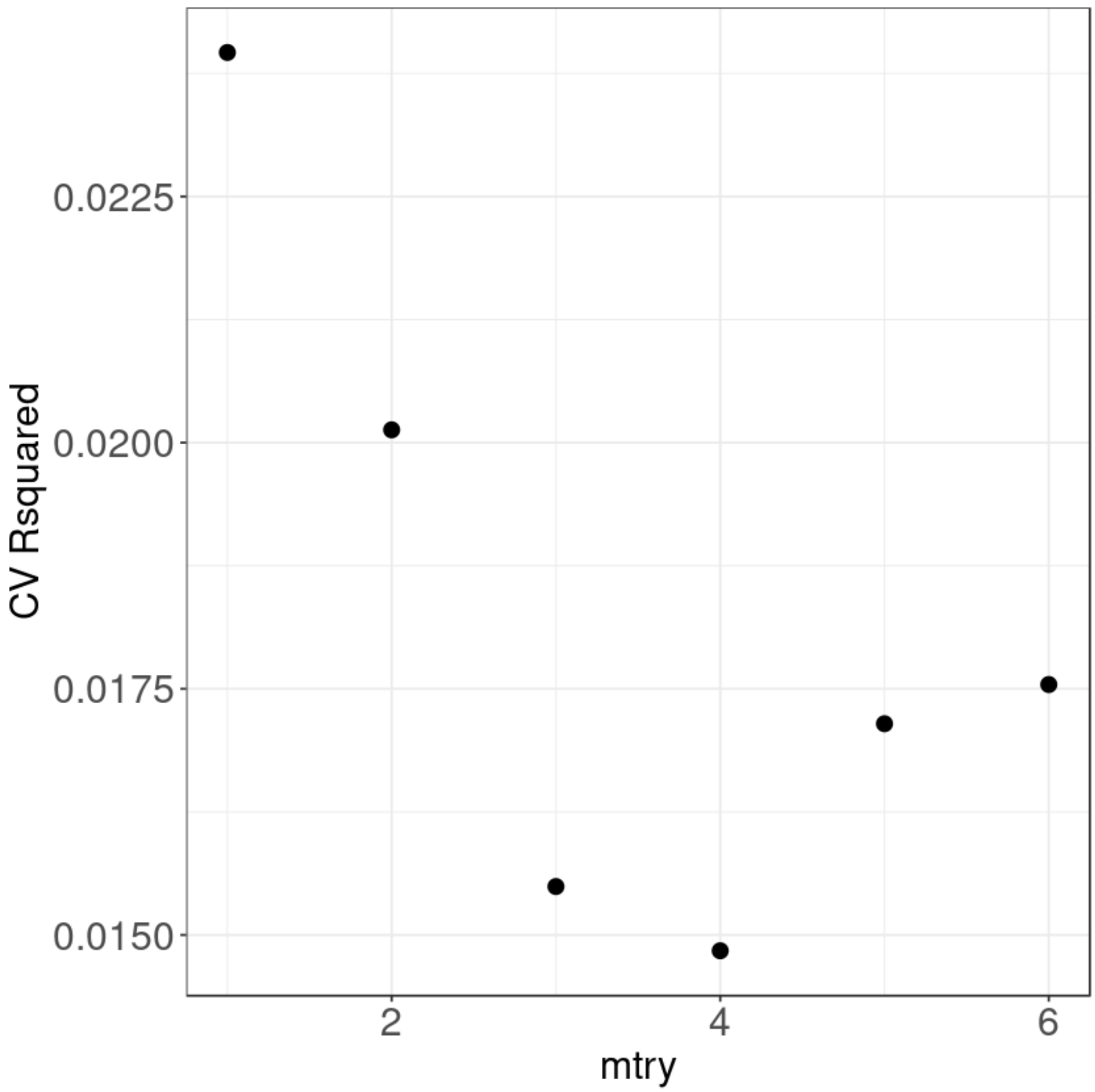
```
train.rf$results
train.rf
best.rf <- train.rf$finalModel
pred.rf <- predict(best.rf, newdata = price_test_data_log.mm)
all_metrics(price_test_data$price_log_diff, pred_lm_log,
        price_train_data$price_log_diff)
```

mtry <int>	RMSE <dbl>	Rsquared <dbl>	MAE <dbl>	RMSESD <dbl>	RsquaredSD <dbl>	MAESD <dbl>
1	0.9868626	0.02396270	0.7021205	0.09754761	0.03461219	0.04148504
2	1.0030523	0.02013034	0.7180743	0.09988699	0.03206543	0.03882143
3	1.0138758	0.01549198	0.7273112	0.10018572	0.02833747	0.03858953
4	1.0172491	0.01483976	0.7298685	0.09965039	0.02697329	0.04055537
5	1.0190250	0.01714539	0.7307562	0.10116483	0.03156220	0.03939837
6	1.0224764	0.01754349	0.7330653	0.10168716	0.03172612	0.04190727

```
Random Forest

1037 samples
   6 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 804, 805, 805, 804, 802
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared    MAE
  1     0.9868626  0.02396270  0.7021205
  2     1.0030523  0.02013034  0.7180743
  3     1.0138758  0.01549198  0.7273112
  4     1.0172491  0.01483976  0.7298685
  5     1.0190250  0.01714539  0.7307562
  6     1.0224764  0.01754349  0.7330653

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 1.
```

1. 1.3321906348193 2. 0.766205419022985 3. 0.0152780706337514

In [51]:
```
ggplot(train.rf$results, aes(x = mtry, y = Rsquared)) + geom_point(size = 3) +
  ylab("CV Rsquared") + theme_bw() + theme(axis.title=element_text(size=18),
  axis.text=element_text(size=18))
```



The best value is 1 for mtry.

```
In [52]: filter_vec = !is.na(pred.rf)
         sum((exp(pred.rf)*price_test_data$label_growth_rate_Price)[filter_vec])
                 /sum(exp(pred.rf)[filter_vec])

         # Baseline
         sum(price_test_data$label_growth_rate_Price[filter_vec])
                 /length(price_test_data$label_growth_rate_Price[filter_vec])
```

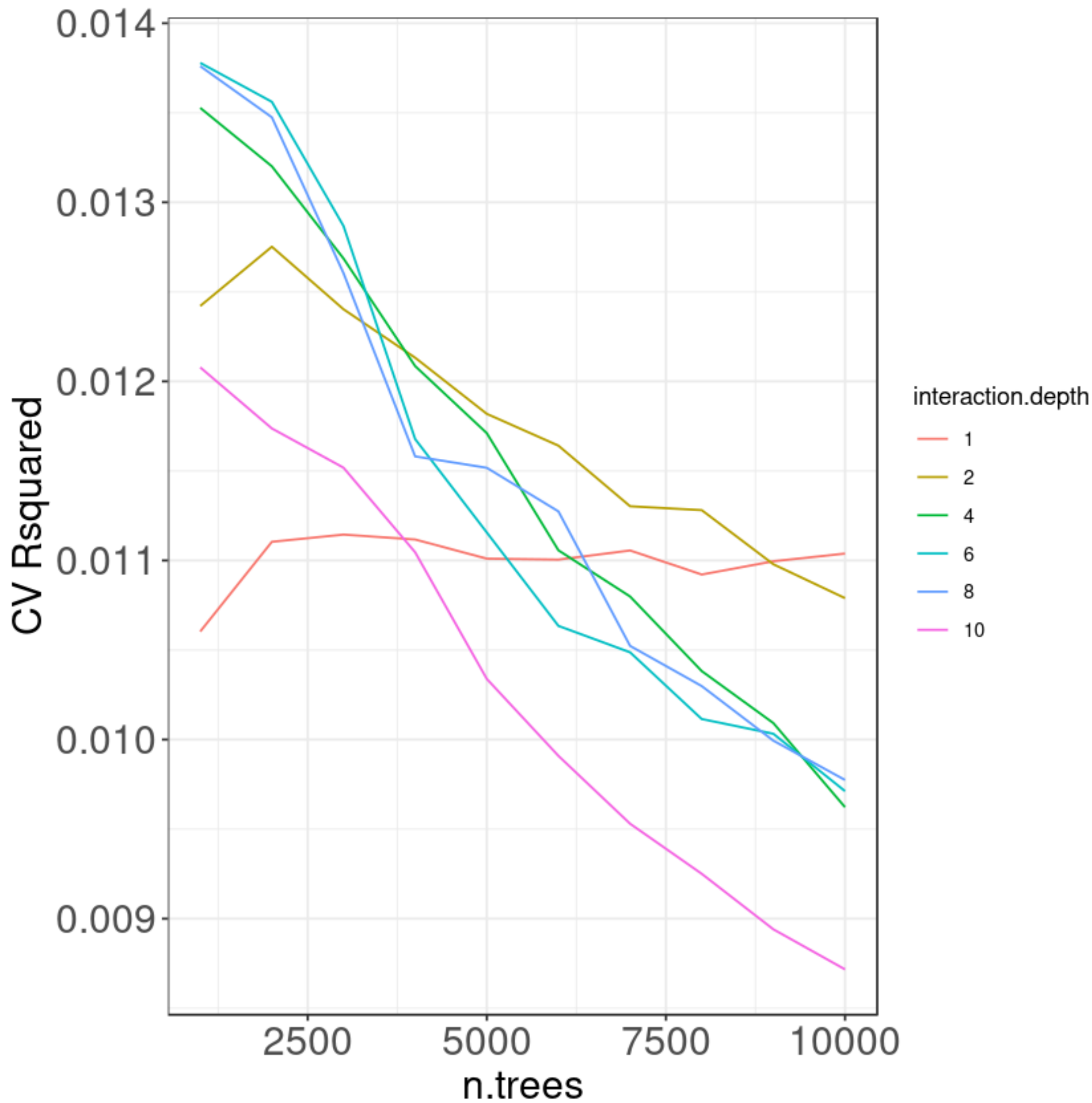
0.318126401502254
0.410949916909122
Unfortunately this model does not beat the baseline.

c) Boosting

We use the same features.

```
In [54]: tGrid = expand.grid(n.trees = (1:10)*1000, interaction.depth = c(1,2,4,6,8,10),
                             shrinkage = 0.001, n.minobsinnode = 10)

         set.seed(42)
         train.boost <- train(price_log_diff ~ X24h + X7d + X200d + Mkt..Cap + Age..mo. +
                              Business,
                              data = price_train_data,
                              method = "gbm",    ## gradient boosting machine
                              tuneGrid = tGrid,
                              trControl = trainControl(method="cv", number=5),
                              distribution = "gaussian",
                              metric = "RMSE",
                              na.action = na.omit)
         train.boost
         best.boost <- train.boost$finalModel

         ggplot(train.boost$results, aes(x = n.trees, y = Rsquared, colour =
           as.factor(interaction.depth))) + geom_line() +
           ylab("CV Rsquared") + theme_bw() + theme(axis.title=element_text(size=18),
           axis.text=element_text(size=18)) +
           scale_color_discrete(name = "interaction.depth")
```

```
Stochastic Gradient Boosting

1037 samples
   6 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 803, 804, 805, 804, 804

Tuning parameter 'shrinkage' was held constant at a value of 0.001

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 1000, interaction.depth =
 6, shrinkage = 0.001 and n.minobsinnode = 10.
```



The best parameters are the biggest ones.

```
In [56]: pred.best.boost <- predict(best.boost, newdata = price_test_data_log.mm,
        n.trees = 10000, interaction.depth=10) # from CV plot
```

```
Warning message in predict.gbm(best.boost, newdata = price_test_data_log.mm, n.trees = 10000, :
“Number of trees not specified or exceeded number fit so far. Using 1000.”
```

It does not want to do it so we need to do it ourselves.

```
In [63]: library(gbm)

         mod.boost <- gbm(price_log_diff ~ X24h + X7d + X200d + Mkt..Cap + Age..mo. + Business,
                          data = price_train_data,
```

```
                                  distribution = "gaussian",
                                  n.trees = 10000,
                                  shrinkage = 0.001,
                                  interaction.depth = 10)

# NOTE: we need to specify number of trees to get a prediction for boosting
pred.boost <- predict(mod.boost, newdata = price_test_data_log.mm, n.trees=10000)
```

In [64]:
```
filter_vec = !is.na(pred.boost)
sum((exp(pred.boost)*price_test_data$label_growth_rate_Price)[filter_vec])
        /sum(exp(pred.boost)[filter_vec])

# Baseline
sum(price_test_data$label_growth_rate_Price[filter_vec])
        /length(price_test_data$label_growth_rate_Price[filter_vec])
```

0.290123603135029
0.410949916909122
This is also very bad compared to the baseline. Too much overfitting!