ALY 6040 Final Project: Draft Report

Students' name: Mohammad Hossein Movahedi, Bharat Goel

Assignment title: Draft Report

Course number and title: ALY6040 71368 Data Mining Applications SEC 09 Fall 2022 CPS [TOR-A-HY]

Term: 202315_A Fall 2022 CPS Quarter First Half

Instructor's name: Hootan Kamran, Ph.D.

Oct 11, 2022,

Introduction

This is the final report for our project. In the previous part (EDA), we cleaned the dataset we got to the last 20 variables.
In this part, we create a model for predicting the stock. We will also answer all the questions we asked in the proposal.

Brief describe of dataset

Since the dataset has a lot of variables, a thorough pre-processing was done by using tree-based algorithm r-part to determine the most important variables in the dataset. Then a correlation matrix was determined in order to remove the highly correlated variable and again the r-part algorithm was ran onto the dataset to finalize the most important variables to work on. All these tasks were performed in the initial report knows and Exploratory Data Analysis. Hence, this report is more focused on applying a technique to determine the best model for our dataset in order to predict the stock prices with the highest accuracy.
Table 1 shows the final selection of variables that we used in this project.
The unit of analytics in this project is best combination of accuray and sensitivity of models that we use to predict. The number of observation is 844 observation and there are 25 variables. One is the name of stock and two are the future value each stock(class) and howmuch that stock performed well in future (AnalystRating)

List of the questions

Below is the list of our initial questions

1. Are there any upper and lower outliers likely to skew results?
2. How effective can the random forest predict the Class variable?
3. Which model can more accurately predict the Class variable?
4. Which model can more accurately predict the analytic rating variable?

Our Approach

The following is the walkthrough of questions and the method we used to answer them

Question 1: Are there any upper and lower outliers likely to skew results?

To answer this question, we used Cook’s Distance method. Since Cook's distance is calculated concerning a certain regression model, only the X variables that are part of the model affect it. Cook's distance calculates each data point (row) impact on the projected outcome.

The cook's distance for each observation I determines how much the observation I affected the fitted values by measuring the change in Ŷ Y^ (fitted Y) for all observations with and without observation i(R-bloggers, 2016).

In the Code in the appendix, we used the cook’s distance method to calculate heavy influence outliers; however, we didn’t do anything with this information since we wanted to use an ML method, and this method is based on GLM, but it was still a good run.

Question 2: What are different sectors' positive and negative price variances?

By group and summarizing the data, we found the answer to this question; the answer is shown in Table 2. As can be seen the highest variance is in book value per share for utilities sector

Question 3: Which model can more accurately predict the Class variable?

Random forest

The first model we used is the random forest. Random forest is generally one of the best ML algorithms for classification problems.it’s aslo wildly used in financial world. The first step to use this algorithm is to split the data into training and testing sub sets and then run the algorithms until it generate a solution . and then look for the best number of variables for each tree and at last create a confusion matrix and look at how good the algorithm is on testing data.

After we follow these steps, created the algorithm and looked at the final result we found out the algorithm is performing poorly. Most of the stock performed well in 2016 and therefor the algorithm has very low sensitivity and by just predicting all stocks will be profitable it has accuracy of 85 percent and above.

This problem is due to the nature of random forest algorithms that they are heavily dependent on training data. However we shoudn’t blame the algorithm for poorly performing in financial world thus no algorithm has been developed that can perform well. There are meny instances of famous hedge funds performed poorly and it shows how hard is navigating the stock market.

However along the way be learned a lot about random forest.

Naïve Bayes Classifier Algorithm

While the result of the random forest was discouraging the Naive Bayes worked pretty well on our dataset giving the situation. The accuracy of the model for both training and testing data was more that 90 percent and the sensitivity of the model for training dataset was around 40 percent while for the test dataset it was around 25 present.

The Naive Bayes classification approach relies on the Bayes Theorem and the presumption of predictor independence. The Naive Bayes model is easy to build and works well with large data sets. If you have a sizable dataset, think about using Naive classification (finnstats, 2021).

The code for this algorithm also can be found in the appendix from line 334 to 366. The sucsess of this algorithm encoreaged us to continue looking for better algorithms .

Support Vector Machine Learning Algorithm

This algorithm failed again.it fail was even more notice able that random forest. The algorithm some how decided to go for 0 sensitivity and went for 85 accuracy without regarding the sensitivity.

Support vector machines, or SVMs, are supervised machine learning algorithms that are mostly used to categorize data into groups. SVM uses a hyperplane, which functions as a decision boundary between the multiple classes, unlike most methods (Lateef, 2019).

k mean clustering

Our attempt in k-mean clustering was kind of hoping that the final clusters have a corelation with the class variable. However this didn’t happen and clusters and class variables didn’t have a strong correlation.

The problem with unsupervised method such as clustering is that all you can do is hope for the best while the algorithm works it’s way.

The code for this alogortim can be found in the appendix from line 411 to 425.

K-NN Classifier

After the k-mean , we tried the k-nn algorthm which is based on same principal as k-mean and is supervised .K Nearest Neighbour is a supervised learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points. the k-nn didn't work well either and had very low sensitivity.

The problem with this algorithm is also being very sensitive to the training data,
The code for this alogortim can be found in the appendix from line 426 to 455.

Question 4: Which model can more accurately predict the analytic rating variable?

The "Analyst Rating" is used as the dependent variable in the first model to determine how much price will change in the following year.
In order to select the best model, the following algorithms were used on the dataset.

- Linear Discriminant Analysis (LDA)
- Classification and Regression Trees (CART)
- k-Nearest Neighbors (KNN)
- Bayesian Generalized Linear Model (GLM)
- Support Vector Machines (SVM)
- Random Forest

The final result of applying all these method shows that although there are differences in accuracy when it comes to predicting the results all of the model can't perform that good.
The kappa test for all of the models doesn't show a significant advantage for the model over random prediction.

The code for this part can be found in appendix form line 457 to 497.

Interpretation

In this project we went through so many different machine learning methods and we tried our best to predict class or analytic rating variable and the models they succeed in doing so however all of models had low sensitivity. Therefore the sensitivity was our biggest problem in this project.all the confusion matrixes show promising accuracy but very low sensitivity. The best model we came up in our project is Naïve Bayes Classifier Algorithm.

Conclusion

This study concentrated on using all widely used methods to choose the optimum model for our dataset in order to forecast stock prices with the highest degree of accuracy and sensitivity. Additionally, we will address every query we raised in the proposal.

Our recommendation for future research in this area is to avoid data cleaning as much as they can since most data cleaning methods are based on parametric statics and normal data however financial data is far from normal by nature and should not be treated as one.

Table and figures

```
> str(finalData2)
'data.frame':	844 obs. of  25 variables:
 $ ebitda                              : num  6.80e+08 7.67e+08 3.08e+08 
1.75e+08 4.25e+08 ...
 $ operating_cash_flow                 : num  5.93e+08 6.06e+08 3.19e+08 
3.48e+08 3.15e+08 ...
 $ price_to_operating_cash_flows_ratio: num  19.86 18.36 7.73 8.4 12.6 ...
 $ price_fair_value                    : num  5.86 5.22 2.84 2.49 3.03 ...
 $ operating_cash_flow_per_share       : num  2.2 2.31 5.96 2.55 1.5 ...
 $ operating_cash_flow_per_share_2     : num  2.2 2.31 5.96 2.55 1.5 ...
 $ pocf_ratio                          : num  19.86 18.36 7.73 8.4 12.6 ...
 $ r_d_to_revenue                      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ free_cash_flow_growth               : num  -0.192 0.159 1.261 -0.374 
0.536 ...
 $ asset_growth                        : num  0.1059 -0.0235 -0.0596 0.0439 
-0.038 ...
 $ book_value_per_share_growth         : num  0.0094 -0.0081 -0.0433 -0.2999 
0.0348 ...
 $ sg_a_expenses_growth                : num  0.0631 0.0319 0.0215 0.112 
0.0085 -0.0978 0.0048 0.177 0.0674 -0.0463 ...
 $ operating_cash_flow_sales_ratio     : num  0.1002 0.1785 0.0615 0.246 
0.0841 ...
 $ stock_based_compensation_to_revenue: num  0.0049 0.0001 0.0013 0.0088 
0.005 0.0065 0.0044 -0.0001 0.0001 0.0065 ...
 $ return_on_equity                    : num  0.192 0.203 0.145 -0.112 0.157
```

```
...
 $ net_profit_margin_2                : num  0.0592 0.1209 0.0221 -0.0945
0.0469 ...
 $ net_income_per_share               : num  1.3 1.565 2.138 -0.979 0.838
...
 $ eps                                : num  1.3 1.57 2.08 -0.98 0.84
-0.046 3.14 0.76 0.65 0.87 ...
 $ cash_per_share_2                   : num  1.0204 1.2586 0.9777 1.7129
0.0359 ...
 $ shareholders_equity_per_share      : num  6.77 7.72 14.77 8.76 5.36 ...
 $ book_value_per_share               : num  6.77 7.72 14.77 8.76 5.36 ...
 $ AnalystRating                      : chr  "Hold" "Hold" "Buy" "Sell" ...
 $ sector                             : chr  "Consumer Defensive" "Consumer
Defensive" "Consumer Defensive" "Consumer Defensive" ...
 $ class                              : int  1 1 1 0 0 0 1 1 1 1 ...
 $ x                                  : chr  "NWL" "CHD" "BIG" "HMHC" ...
```

Table 1

sector	Basic Materials	Communication Services	Consumer Cyclical	Consumer Defensive	Energy	Financial Services	Healthcare	Industrials	Real Estate	Technology	Utilities
ebitda_var	3.26878E+16	20016088355333332	53429197628206584	35729539761844124	67323690075664664	18746532398670052	33026997352391124	35180388449344360	44223467825209752	25318804008269588	35496441453567888
operating_cash_flow_var	31461800835801104	8199503413952381	30837240967551588	25672949520185380	5.96681E+16	12774855650476522	20640740945427316	21360964855717780	22894393987460460	21666503839799212	19202619140340380
price_to_operating_cash_flows_ratio_var	46.07976462	30.49011238	36.2795882	70.6581761	41.77466585	49.1105359	43.03666649	45.15482904	55.04385715	58.01989276	13.46572868
price_fair_value_var	1.513281043	1.842282905	1.911980405	3.120815928	0.935980681	0.5832584244	2.055783513	2.081778075	1.395338059	1.662557332	0.4463887024
operating_cash_flow_	4.378697629	2.437090119	2.948142993	3.693242623	4.161712368	1.91007556	3.11650889	3.18443106	1.896669146	3.574903436	4.383277671

per_share_var											
operating_cash_flow_per_share_2_var	4.378697629	2.437090119	2.948142993	3.693242623	4.161712368	1.91007556	3.11650889	3.18443106	1.896669146	3.574903436	4.383277671
pocf_ratio_var	46.07976462	30.49011238	36.2795882	70.6581761	41.77466585	49.1105359	43.03666649	45.15482904	55.04385715	58.01989276	13.46572868
r_d_to_revenue_var	0.0001334760213	0.0004957347619	0.0002299054803	6.67E-05	3.98E-05	0	0.000827849697	0.0002072305685	0	0.0006395904036	4.09E-07
free_cash_flow_growth_var	0.5559283887	0.175072669	0.4395053166	0.5829819457	0.9210554381	0.249570936	0.4839582733	0.5032273753	0.645036301	0.5592361616	0.6935746321
asset_growth_var	0.01068261587	0.007192598095	0.009080347042	0.008696909218	0.01448351103	0.008938435988	0.01043770924	0.01239952891	0.01086888803	0.01265157385	0.002711265649
book_value_per_share_growth_var	0.01584583363	0.02420936905	0.01665207177	0.01856563975	0.01742727846	0.00890442299	0.01449943772	0.01599320079	0.01317289504	0.01480310648	0.007576970996
sg_a_expenses_growth_var	0.02282568638	0.05070628476	0.01191484381	0.01029131325	0.03313877883	0.008756953654	0.01036434652	0.01442820496	0.01592881278	0.01765305919	0.003353833506
operating_cash_flow_sales_ratio_var	0.0111105611	0.02586140594	0.004023447852	0.005666303499	0.02863134462	0.01889435193	0.005566467347	0.0141319281	0.02150018746	0.0081276479	0.005425897541
stock_based_compensation_to_revenue_var	1.51E-05	0.0001762080952	3.10E-05	2.93E-05	0.000172922849	8.55E-05	8.88E-05	4.28E-05	0.0001319268033	0.0001314556144	8.16E-06
return_on_equ	0.0135000124	0.0231627090	0.0114648925	0.0109047136	0.0129148517	0.0024982258	0.0114398951	0.0110579928	0.0060782484	0.0087852458	0.0010559004

ity_var	8	5	3	2	4	17	9	2	74	09	55
net_profit_margin_2_var	0.00899508118	0.004857096784	0.00164282032	0.003841280632	0.008592940397	0.008975162593	0.006272251893	0.00620659468	0.0224378364	0.003729903112	0.002862842766
net_income_per_share_var	2.372445546	0.2787469467	1.397695756	1.36300559	2.590269783	0.747028688	1.65768709	1.83914836	1.054429413	1.96293094	0.8243578517
eps_var	2.358882393	0.2713142857	1.458951996	1.393337023	2.247375524	0.7273558719	1.659510517	1.812159392	1.081792999	1.961379588	0.9036244663
cash_per_share_2_var	6.189235621	0.932483479	5.297064918	6.852683394	10.26085306	9.002156611	4.092458787	5.889966494	3.657538621	7.948178249	1.70653669
shareholders_equity_per_share_var	68.32492256	64.95515101	76.96912087	99.1322344	84.32634642	57.66323505	71.40043881	66.00146042	70.06833534	96.09746027	90.46224944
book_value_per_share_var	68.77712693	64.95507162	77.08619264	99.13227283	78.96682466	57.66363372	71.39935537	66.09106347	70.06891509	95.91241045	100.9960649
class_var	0.0664160401	0.2380952381	0.1794139427	0.1664904863	0.1994301994	0.0517462194	0.2495543672	0.1306168831	0.1261237441	0.1621621622	0.04545454545

Table 2

Refrences

Classifier evaluation with imbalanced datasets. (2015). Basic evaluation measures from the confusion matrix. [online] Available at: https://classeval.wordpress.com/introduction/basic-evaluation-measures/ [Accessed 17 Oct. 2022].

Erin Sovansky Winter (2022). Chapter 4: apply Functions. [online] Uic.edu. Available at: https://ademos.people.uic.edu/Chapter4.html [Accessed 17 Oct. 2022].

finnstats (2021). Naive Bayes Classifier in Machine Learning» Prediction Model» finnstats. [online] finnstats. Available at:

https://finnstats.com/index.php/2021/04/08/naive-bayes-classification-in-r/ [Accessed 16 Oct. 2022].

Lateef, Z. (2019). Data Science with R Programming Certification Training Course. [online] Edureka. Available at: https://www.edureka.co/blog/support-vector-machine-in-r/#:~:text=SVM%20(Support%20Vector%20Machine)%20is,boundary%20between%20the%20various%20classes. [Accessed 16 Oct. 2022].

R-bloggers. (2016). Outlier detection and treatment with R | R-bloggers. [online] Available at: https://www.r-bloggers.com/2016/12/outlier-detection-and-treatment-with-r/ [Accessed 11 Oct. 2022].

Statmethods.net. (2017). Quick-R: Tree-Based Models. [online] Available at: https://www.statmethods.net/advstats/cart.html [Accessed 17 Oct. 2022].

Appendix

```r
install.packages('tidyverse')
install.packages('MASS')
install.packages('car')
install.packages('e1071')
install.packages('caret')
install.packages('carTools')
install.packages('cowplot')
install.packages('pROC')
install.packages('ggcorrplot')
install.packages('corrplot')
install.packages('dplyr')
install.packages('janitor')
install.packages("mlbench")
install.packages('repr')
install.packages("caTools")
install.packages("randomForest")
install.packages('effects')
# Loading libraries
library(repr)
library(tidyverse)
library(MASS)
library(car)
library(e1071)
```

```
library(caret)
library(carTools)
library(cowplot)
library(pROC)
library(ggcorrplot)
library(dplyr)
library(janitor)
library(mlbench)
library(data.table)
library(matrixStats)



# loading the training dataset
d15 <- read.csv("2015_Financial_Data.csv")
#looking at the dataset
dim(d15)
summary(d15)


#Gather missing data and plot graph to visualize missing data within
variables
data_miss <- d15 %>% summarise_all(funs(sum(is.na(.))/n()))
data_miss <- gather(data_miss, key = "variables", value =
"percent_missing")
data_miss2=subset(data_miss,data_miss$percent_missing>0.1)
ggplot(data_miss2, aes(x = reorder(variables, percent_missing), y =
percent_missing)) +
  geom_bar(stat = "identity", fill = "red", aes(color = I('white')), size =
0.3)+
  xlab('variables')+
  coord_flip()+
  theme_bw()

##Deleting variables from original dataset
data2=subset(d15,select =
-c(43,53,76,77,78,82,83,84,85,87,88,90,91,96,97,99,100,101,

103,105,112,114,120,127,128,129,130,132,143,146,148,149,150,

151,152,153,155,159,172,173,174,175,186,201,202,204,205,207,208,210,211,213
,214))
data3=subset(data2,select = -c(80))
```

```
data4=subset(data3,select =
-c(15,19,21,33,34,38,45,47,49,50,51,94,103,126,150))

#Modifying variable names
data4<-data4 %>% clean_names()
colnames(data4)

#Categorizing companies as small, medium, or large cap
market_cap_cat<-cut(data4$market_cap,breaks = c(0,1e+9,1e+10,1e+20),labels
=c("Small","Mid","Large") )
data4$market_cap_cat=market_cap_cat
table(market_cap_cat)
data5=subset(data4,select = -c(100,138,136))

#Performing structural adjustments to the data
data5$x1=as.character(data5$x1)
data5$market_cap_cat=as.factor(data5$market_cap_cat)
data5$class=as.factor(data5$class)
data5$sector=as.factor(data5$sector)

#eliminate colinearity/non-essential variables
fit01 = lm(x2016_price_var~.- x - sector - class - market_cap_cat,data5)
summary(fit01)

#Create new dataset elminating "N/A" variables
data6=subset(data5,select =
-c(11,18,23,28,29,67,68,86,87,88,93,94,100,101,102,104,
                               105,107,110,124,125,126,128,129))

#Create a new variable to categorize price variance into broader baskets
summary(data4$x2016_price_var)
AnalystRating<-cut(data4$x2016_price_var,breaks =
c(-100,-50,-5.17,17.28,40.57,4000),
                   labels =c("Strong Sell","Sell","Hold","Buy","Strong
Buy") )
AnalystRating[1:10]
data6$AnalystRating=AnalystRating

#Use tree based algorithm r-part to determine the most important variables
in the dataset
rpartMod<-train(AnalystRating~.- x - sector - class -
market_cap_cat-x2016_price_var,data = data6,method="rpart",na.action =
na.exclude)
```

```
rpartImp<-varImp(rpartMod)
print(rpartImp)

#Putting selected variables into a dataset
VariableCorr2=subset(data6,select = c('operating_cash_flow_sales_ratio'
,'operating_cash_flow','stock_based_compensation_to_revenue',

'return_on_equity','net_profit_margin_2','net_income_per_share','eps','cash
_per_share_2',

'shareholders_equity_per_share','book_value_per_share' ))
colnames(VariableCorr2)
summary(VariableCorr2)


#Check that selected variables are not heavily correlated
corr2<-round(cor(VariableCorr2),1)
ggcorrplot(corr2,lab = TRUE)

finalData=subset(data6,select = c('x','operating_cash_flow_sales_ratio'
,'operating_cash_flow','stock_based_compensation_to_revenue',

'return_on_equity','net_profit_margin_2','net_income_per_share','eps','cash
_per_share_2',

'shareholders_equity_per_share','book_value_per_share',"AnalystRating",'sec
tor','class'))
summary(finalData)
colnames(finalData)
# choosing the best variables based on random forest to make final variable
selection better
# Loading package
library(caTools)
library(randomForest)
# Splitting data in train and test data
d15[is.na(d15)] <- 0
d15 <- subset(d15, select = - c(X2016.PRICE.VAR....))
split <- sample.split(d15, SplitRatio = 0.7)
split

train <- subset(d15, split == "TRUE")
test <- subset(d15, split == "FALSE")
```

```
# Fitting Random Forest to the train dataset
set.seed(100)  # Setting seed
classifier_RF = randomForest(x = train[-224],
                             y = train$Class,
                             ntree = 100)
# Predicting the Test set results
y_pred = predict(classifier_RF, newdata = test[-224])

# Plotting model
plot(classifier_RF)

varImpPlot(classifier_RF)
# Importance plot
k <- importance(classifier_RF)

# selecting important 10
df <- enframe(k)
df<-df[order(df$value,decreasing = TRUE),]
Names <- c(head(df$name, n = 15L))
varforest  <- d15[ , names(d15) %in% Names]
varforest<-varforest %>% clean_names()

#making final data better
finalData <- left_join(varforest,finalData)
finalData=subset(finalData,select = -c(7))

#Replace missing numerical values to take care of outliers
dt <- data.table(finalData[-1])
indx <- sapply(dt, \(x) !(x %in% boxplot(x, plot=FALSE)$out))
n <-dt[as.logical(rowProds(indx))]
n <- as.data.frame(n)
summary(n)
#Replace missing values of numeric variables
for(i in 1:ncol(n)){
  n[is.na(n[,i]), i] <- mean(n[,i], na.rm = TRUE)
}
n <- n %>%
  na.omit()
summary(n)

finalData2 <- left_join(n,finalData)
finalData2=subset(finalData2,select = -c(6))
finalData2 <- finalData2 %>%
```

```
  na.omit()

#EDA
options(repr.plot.width = 17, repr.plot.height = 10)
ggplot(finalData2, aes(x=sector,fill=AnalystRating))+ geom_bar()+
theme_bw()

ggplot(finalData2, aes(x=AnalystRating, y=eps, fill=AnalystRating)) +
geom_violin()+
  geom_boxplot(width=.1, fill="white") + labs(title="EPS")

ggplot(finalData2, aes(x=AnalystRating, y=return_on_equity,
fill=AnalystRating)) + geom_violin()+
  geom_boxplot(width=0.1, fill="white") + labs(title="Total Return on
Equity")

ggplot(finalData2, aes(x=AnalystRating, y=net_profit_margin_2,
fill=AnalystRating)) + geom_violin()+
  geom_boxplot(width=0.1, fill="white") + labs(title="Total Net Profit
Margin")

ggplot(finalData2, aes(x=AnalystRating, y=net_income_per_share,
fill=AnalystRating)) + geom_violin()+
  geom_boxplot(width=0.1, fill="white") + labs(title="net income per
share")

ggplot(finalData2, aes(x=AnalystRating, y=operating_cash_flow_sales_ratio,
fill=AnalystRating)) + geom_violin()+
  geom_boxplot(width=0.1, fill="white") + labs(title="operating cash flow
sales ratio")


finalData2 %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
  facet_wrap(~ key, scales = "free") +   # In separate panels
  geom_density()

#finalData2
write.csv(finalData2,"~/Data/finalData2.csv", row.names = FALSE)
finalData2 <- read.csv('finalData2.csv')
```

```
names(finalData2)


#Question 1 : Are there any upper and lower outliers likely to skew
results?
## Cook's Distance Method
mod <- glm(class ~ .-AnalystRating-x, data = finalData2,family=binomial)
par(mfrow = c(2, 2))
library(effects)
plot(allEffects(mod))
summary(mod)
dev.off()
cooksd <- stats:::cooks.distance.glm(mod)
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  #
plot cook's distance
abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd,
na.rm=T),names(cooksd),""), col="red")  # add labels

influential <- cooksd[(cooksd > (1 * mean(cooksd, na.rm = TRUE)))]
influential
ci <- ifelse(cooksd>4*mean(cooksd, na.rm=T),1,0)  # influential row numbers
table(ci)

names_of_influential <- names(influential)
outliers <- finalData2[names_of_influential,]
finalData2_without_outliers <- finalData2 %>% anti_join(outliers)
modn <- glm(class ~ .-AnalystRating-x, data =
finalData2_without_outliers,family=binomial)
par(mfrow = c(2, 2))
plot(modn)

# Question 2: What are different sectors' positive and negative price
variances?
k <-finalData2 %>%
  group_by(sector) %>%
  summarise(across(where(is.numeric), list( var = var)))
write.csv(k,"~/Data/k.csv", row.names = TRUE)

#Question 3
## random forest algorithm
#Installing and loading nessesarry libraries
install.packages('pacman')
```

```
install.packages('tidyverse')
install.packages('tidymodels')
install.packages('leaps')
install.packages('glmnet')
install.packages('BMA')
install.packages('janitor')
install.packages("randomForest")
install.packages('effects')

library(tidyverse)
library(tidymodels)
library(leaps)
library(pacman)
library(glmnet)
library(BMA)
library(janitor)
library(caTools)
library(randomForest)

#loading the cleaned dataset

dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating))
dim(dt)

# running random forest for the class variable
# spiting data into test and train

# Splitting data in train and test data
set.seed(321)  # Setting seed
split <- sample.split(dt, SplitRatio = 0.8)
split

train <- subset(dt, split == "TRUE")
test <- subset(dt, split == "FALSE")

# Fitting Random Forest to the train dataset
set.seed(123)  # Setting seed
classifier_RF = randomForest(class~.,
                             data = train,
                             mtry = 11,
                             ntree = 2000)
```

```
classifier_RF

# finding mtry
a=c()
i=5
set.seed(123)  # Setting seed
for (i in 3:23) {
  model3 <- randomForest(class ~ ., data = train, ntree = 100, mtry = i,
importance = TRUE)
  predValid <- predict(model3, newdata = test[-23])
  a[i-2] = mean(predValid == test$class)
}
a
plot(3:23,a)

# adjusted forest
set.seed(123)  # Setting seed
classifier_RF = randomForest(class~.,
                             data = train,
                             mtry = 11,
                             ntree = 2000)

classifier_RF


# Predicting the Test set results
y_pred = predict(classifier_RF, newdata = test[-23])

# Confusion Matrix
mtx = table(test[,23], y_pred)
#Calculating accuracy

confusionMatrix(mtx)

# Plotting model
plot(classifier_RF)

varImpPlot(classifier_RF)
# Importance plot
k <- importance(classifier_RF)
k
```

```
a=c()
i=5
set.seed(123)  # Setting seed
for (i in 3:23) {
  model3 <- randomForest(class ~ ., data = train, ntree = 2000, mtry = i,
importance = TRUE)
  predValid <- predict(model3, newdata = test[-23])
  a[i-2] = mean(predValid == test$class)
}
a
plot(3:23,a)

##Naive Bayes Classifier
#install packages
install.packages('naivebayes')
install.packages('psych')
library(naivebayes)
library(psych)
# loading cleaned data
#loading the cleaned data set

dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating,x))
dim(dt)
xtabs(~class+sector, data = dt)
# all the ranks are than 5 which satisfy the method needs
#create train and test data sets for training the model and testing
set.seed(1234)
ind <- sample(2, nrow(dt), replace = T, prob = c(0.8, 0.2))
train <- dt[ind == 1,]
test <- dt[ind == 2,]
#Naive Bayes Classification in R
model <- naive_bayes(class ~., data = train, usekernel = T)
model
plot(model)
p <- predict(model, train, type = 'prob')
p1 <- predict(model, train)
tab1 <- table(p1, train$class)
confusionMatrix(tab1)

#testing
p2 <- predict(model, test)
```

```
tab2 <- table(p2, test$class)
confusionMatrix(tab2)

#Support Vector Machine
##installing packages
install.packages('caret')
library(caret)
#reading the dataset
dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating,x))
dim(dt)
# spliting data into training and testing subsets
intrain <- createDataPartition(y = dt$class, p= 0.8, list = FALSE)
training <- dt[intrain,]
testing <- dt[-intrain,]
# train control.this will control all the computational overheads
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
# train() method
svm_Linear <- train(class ~., data = training, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)
svm_Linear
test_pred <- predict(svm_Linear, newdata = testing)
p <-table(test_pred, testing$class)
confusionMatrix(p)
# tuning of an SVM classifier with different values of C
grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5,
1.75, 2,5))
svm_Linear_Grid <- train(class ~., data = training, method = "svmLinear",
                         trControl=trctrl,
                         preProcess = c("center", "scale"),
                         tuneGrid = grid,
                         tuneLength = 10)
svm_Linear_Grid
plot(svm_Linear_Grid)

#k mean clustering
##installing packages
install.packages('factoextra')
library(factoextra)
#reading the dataset
```

```
dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating,x))
dim(dt)
#compute k-means
df <- dt %>%
  keep(is.numeric)%>%
  scale()
df
set.seed(123)
km.res <- kmeans(df, 2, nstart = 25)
fviz_nbclust(df, kmeans, method = "wss") +
  geom_vline(xintercept = 5, linetype = 2)
aggregate(dt, by=list(cluster=km.res$cluster), mean)
fviz_cluster(km.res ,data = df)
dd <- cbind(dt, cluster = as.factor(km.res$cluster))
head(dd)
p <-table(dd$cluster,dd$class)
p
#K-NN Classifier
install.packages("class")
library(class)
#loading data
#reading the dataset
dt <- read.csv('finalData2.csv')
dt$class<- as.factor(dt$class)
dt <- subset(dt, select = -c(AnalystRating,x))
dim(dt)
#spliting the data
# Splitting data into train
# and test data
set.seed(1234)
split <- sample.split(dt, SplitRatio = 0.7)
train_cl <- subset(dt, split == "TRUE")
test_cl <- subset(dt, split == "FALSE")
# Feature Scaling
train_scale <- train_cl %>%
  keep(is.numeric)%>%
  scale()
test_scale <- test_cl %>%
  keep(is.numeric)%>%
  scale()
# Fitting KNN Model
```

```
# to training dataset
classifier_knn <- knn(train = train_scale,
                      test = test_scale,
                      cl = train_cl$class,
                      k = 1)
confusionMatrix(table(test_cl$class, classifier_knn))

#Question 4
finalData2 <- read.csv('finalData2.csv')

#Asses which model can most accurately predict price variance

#Pick the best model
control = trainControl(method="cv", number=10)
metric = "Accuracy"

# Linear Discriminant Analysis (LDA)
set.seed(138)
fit.lda = train(AnalystRating~.- x - class , data=finalData2, method="lda",
metric=metric, trControl=control,na.action = na.pass)

# Classfication and Regression Trees (CART)
set.seed(138)
fit.cart = train(AnalystRating~.- x - class, data=finalData2,
method="rpart", metric=metric, trControl=control)

# k-Nearest Neighbors (KNN)
set.seed(138)
fit.knn = train(AnalystRating~.- x - class, data=finalData2, method="knn",
metric=metric, trControl=control)

# Bayesian Generalized Linear Model
set.seed(138)
fit.logi = train(AnalystRating~.- x - class, data=finalData2,
method="bayesglm", metric=metric, trControl=control)

# Support Vector Machines (SVM)
set.seed(138)
fit.svm = train(AnalystRating~.- x - class, data=finalData2,
method="svmRadial", metric=metric, trControl=control)

# Random Forest
set.seed(138)
```

```
fit.rf = train(AnalystRating~.- x - class, data=finalData2, method="rf",
metric=metric, trControl=control)


# Select Best Model
# summarize accuracy of models
results = resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn,
logi=fit.logi, svm=fit.svm, rf=fit.rf))
summary(results)
```