CASE 2 - eLAB Business Case

24 May 2023

Authors:

- Cornelius Kötting
- Jakub Czubaszek
- Tshiamo Kgosidiile
- Ajay Rayudu

Tutor: Hugo Schyns

1. Introduction

Welcome to the fascinating intersection of finance and technology – where data science, machine learning, and financial markets converge. In the past, financial decisions rested heavily on experience, intuition, or simple strategies. Today, however, we exist in a realm where data is the new oil, transforming the finance sector into a well-oiled machine of precise predictions and strategic insights. What brought about this seismic shift? The answer lies in the power of data science. An intriguing blend of different disciplines, data science employs an array of scientific methods and algorithms, all with one common aim – to draw knowledge from the colossal amounts of data that surround us. In finance, this means unraveling the intricate patterns, elusive trends, and hidden correlations buried within financial data. The insights we glean from this process empower us to predict market changes, assess potential investments, and steer clear of potential financial pitfalls. At the heart of data science sits predictive modelling, an impressive tool in the arsenal of any financial analyst. These models, akin to crystal balls of the digital world, utilize historical data to predict future outcomes. The magic behind these models is machine learning - a subset of artificial intelligence, which, like a skilled apprentice, learns from data and refines its predictions over time. Adaptable and intuitive, machine learning models provide valuable predictions, unfazed by market changes or unseen data. Now, we turn our attention to the central theme of our assignment – the Price-Earnings (PE) ratio. A bellwether of the financial world, the PE ratio serves as an indicator of a company's market value and is often used as a benchmark to determine if a stock is fairly priced. By developing a predictive model for PE ratios, we stand to gain invaluable insights into potential investment prospects and future market trends. The evolution of the financial sector through data science, predictive modelling, and machine learning is a testament to their influence and importance. These technologies, like master keysmiths, have crafted a new paradigm in financial decision-making. They've honed our forecasting abilities, reduced our susceptibility to human error, and shed light on the complex mechanics of market dynamics. As we delve deeper into the intricacies of financial data science in this assignment, we'll uncover how predictive models, particularly concerning PE ratios, aid in interpreting and forecasting the rhythmic dance of financial markets.

2. Data import

```
 data <- read.csv("/Users/cornelius/Downloads/compu.csv", header = TRUE)
data_testing <- read.csv("/Users/cornelius/Downloads/computestpublic.csv", header = TRUE)
```

3. Data cleaning

Before any meaningful analysis can conducted, we need to clean the dataset to ensure that it is ready for model building later on. Since we had some missing data in the columns *prcc_c* and *prcc_f* we opted to go through the data and choose *prcc_c*, unless this was NA, then we choose *prcc_f*. Furthermore, we opted to remove all observations where we had missing data for the variables *ni* and *csho*, as these are needed later on. Moreover we also purged all observations where *at* & *csho* were smaller or equal to zero.

```
 #Filter out rows where prcc_c and prcc_f are both NA and at & csho > 0
data <- data %>% filter(!is.na(prcc_c) | !is.na(prcc_f),
                        !is.na(ni),
                        at > 0,
                        !is.na(csho),
                        csho > 0)

#Removing NA and negative values from testing set
data_testing <- data_testing %>% filter(!is.na(ni),
                        at > 0,
                        !is.na(csho),
                        csho > 0)
```

4. Variables of interest

After some preliminary data pre-processing, it is now time to create some new variables that will be used to construct our predictive models later. The variables to be created can be seen below:

- *MTB*: Market-to-book ratio, which informs us whether a firm is labelled growth or value.
- *size*: Firm size is often measured as the log of total assets.
- *EPS*: Earnings per share
- *PE*: Price earnings ratio

- *inv*: Investments, measured as the change in firm size compared to last year.
- *ROA*: Return on Assets.
- *ROE*: Return on Equity

Additionally, we also filtered the dataset to remove small firms (defined as having a market value < 10), as smaller businesses often exhibit more volatility, have less data available and could skew the model's results and therefore lessen its predictive ability.

```
 #Creating variables of interest
data <- data %>%
  arrange(gvkey, fyear) %>%
  group_by(gvkey) %>%
  mutate(price = ifelse(is.na(prcc_c), prcc_f, prcc_c),
         mktval = price*csho,
         size = log(at),
         EPS = ni/csho,
         PE = price/EPS,
         bkval = ceq,
         bkval = ifelse(is.na(bkval), at-lt, bkval),
         bkval = ifelse(is.na(bkval), bkvlps*csho, bkval),
         MTB = mktval/bkval,
         last_year_size = lag(size),
         inv = size - last_year_size,
         ROA = ni/at,
         ROE = ni/bkval) %>%
  filter(mktval > 10) #Filtering out small firms

data_testing <- data_testing %>%
  arrange(gvkey, fyear) %>%
  group_by(gvkey) %>%
  mutate(size = log(at),
         last_year_size = lag(size),
         inv = size - last_year_size,
         ROA = ni/at) %>% select(c(gvkey, fyear, ROA, inv, size))
```

5. Eploratory data analysis

In the next part of this investigation, exploratory data analysis will be conducted. As we found significant outliers in our dataset, we will transform logarithm transformation to better understand the variables in the data and their relationships. This transformation will only be applied to the data used in the exploratory data analysis, as we found that our models perform better on winsorized independent variables.

```
 #Aggregate data for later visualizations
data_2015_2016 <- data %>%
  filter(fyear%in% c(2015, 2016)) %>%
  select(gvkey, fyear, MTB, ROA, ROE, PE) %>% group_by(fyear)

#Perform log transformation and replace missing values with log(1)
data_2015_2016 <- data_2015_2016 %>%
  mutate(
    MTB = log(replace_na(MTB, 1)),
    ROA = log(replace_na(ROA, 1)),
    ROE = log(replace_na(ROE, 1)),
    PE = log(replace_na(PE, 1))
  )
```

```
diagnose_numeric(select(filter(data_2015_2016, fyear == 2015), -c(gvkey, fyear))) %>% flextable()
```

variables	min	Q1	mean	median	Q3	max	zero	minus	ou
fyear	2,015.000000	2,015.00000000	2,015.0000000	2,015.0000000	2,015.000000	2,015.000000	0	0	
MTB	-3.267388	0.08130172	0.7695808	0.6221345	1.311776	14.153891	0	1,027	
ROA	-11.578844	-4.41177680	-3.4627531	-3.2894979	-2.612469	7.597111	0	3,331	
ROE	-10.203583	-2.69164366	-2.0423277	-2.2113113	-1.571716	7.328370	0	3,300	
PE	-2.087192	2.53923156	3.0116487	2.9066166	3.365196	11.540953	0	9	

```
 diagnose_numeric(select(filter(data_2015_2016, fyear == 2016), -c(gvkey, fyear))) %>% flextable()
```

variables	min	Q1	mean	median	Q3	max	zero	minus	outl
fyear	2,016.000000	2,016.0000000	2,016.0000000	2,016.0000000	2,016.000000	2,016.000000	0	0	
MTB	-3.529042	0.2245828	0.8713328	0.7167077	1.355874	13.254366	0	743	1
ROA	-10.375106	-4.4748062	-3.5089088	-3.3320876	-2.621853	5.901430	0	3,265	
ROE	-8.799293	-2.7148952	-2.0939331	-2.2456519	-1.581860	7.230853	0	3,249	3
PE	-1.546143	2.6548564	3.1306627	3.0225325	3.446909	10.161791	0	17	2

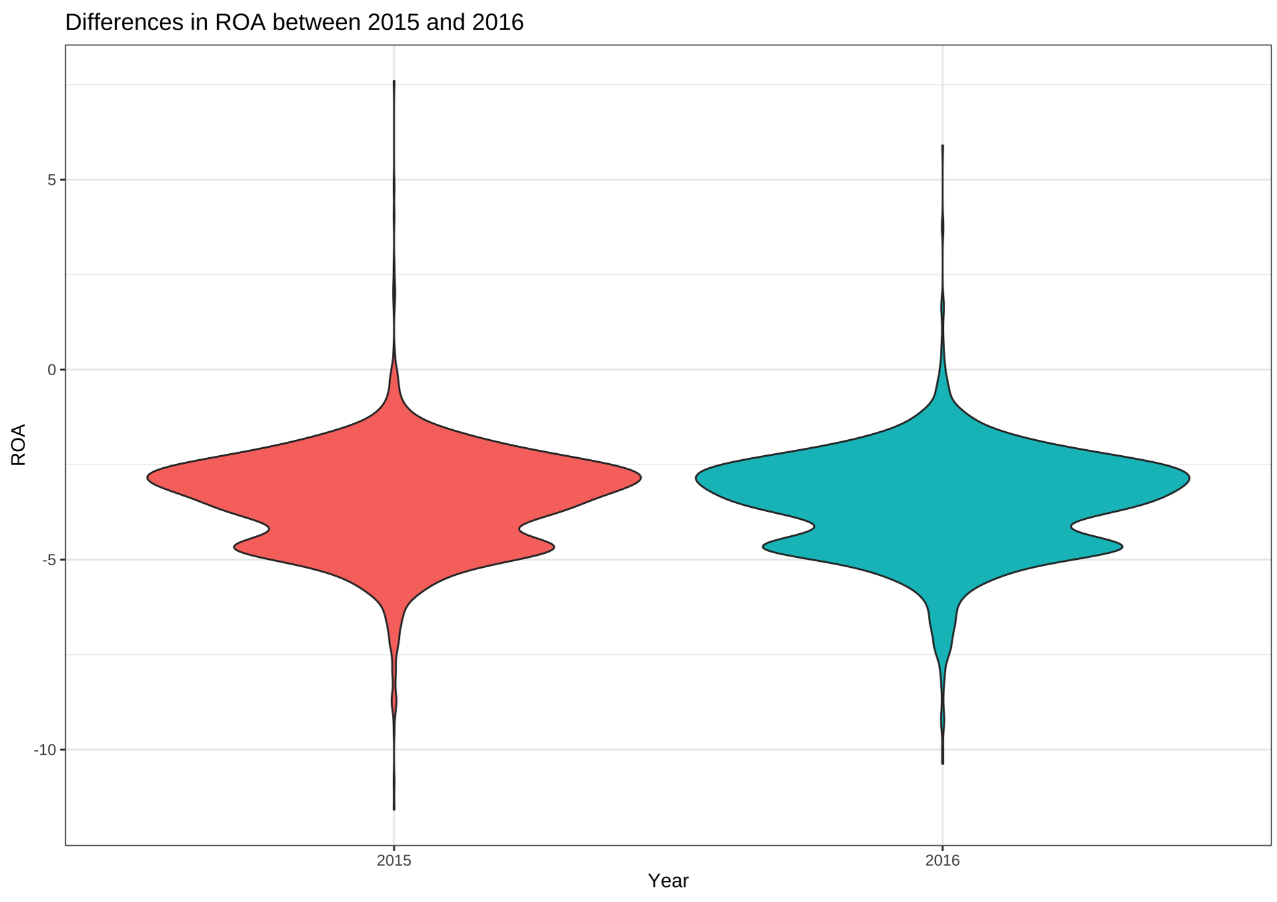
The tables above display summary statistics for our variables of interest first for the year 2015 and then 2016. As we can see the statistics only differ slightly between the two years, which is consistent with common sense, as financials rarely change significantly across a large number of firms except in the event of a crisis.

5.1 Visualizations

To better understand the differences in the dataset between the two fiscal years, we will visualize some relationships in the next part.

5.1.1 Violin plot on ROS in 2015 & 2016

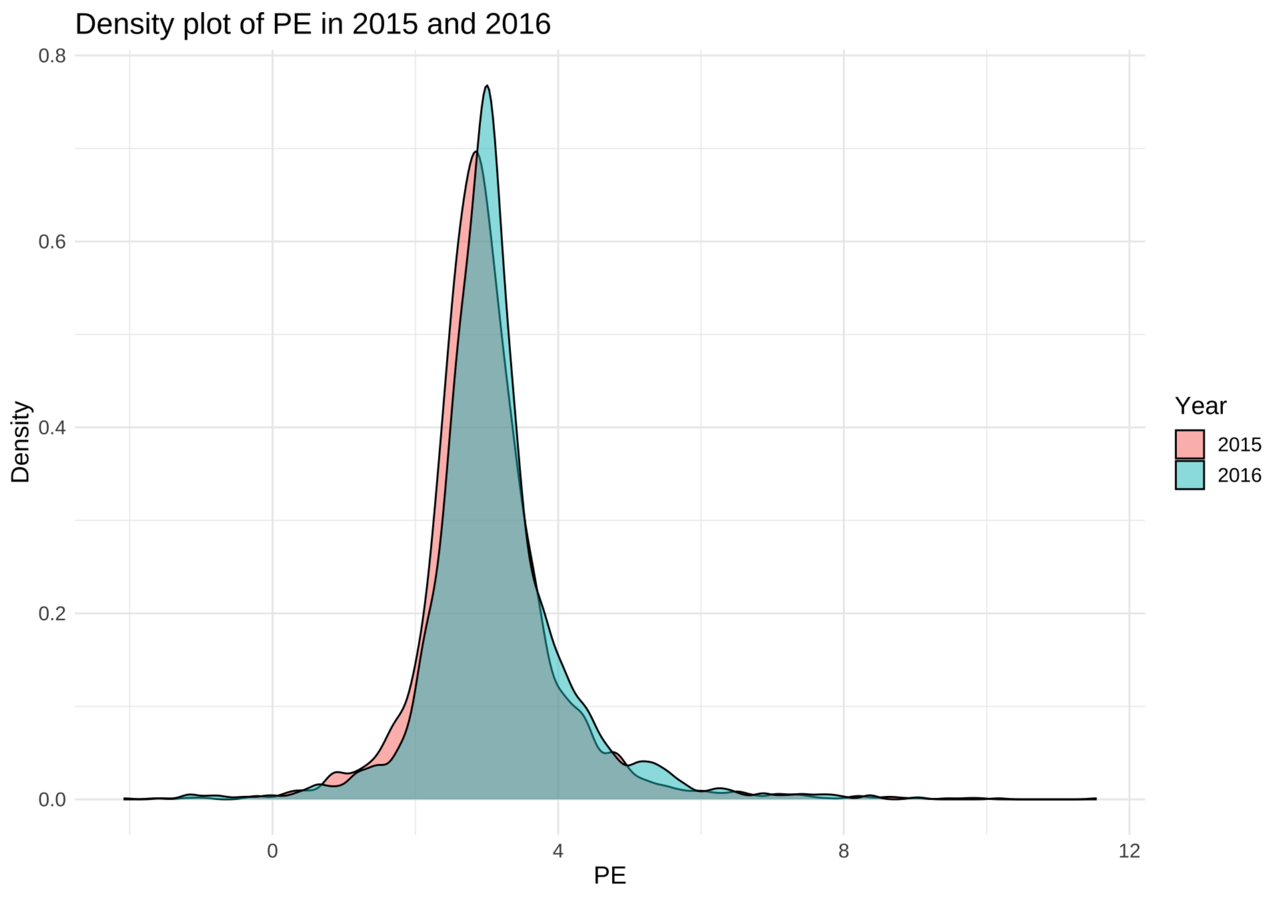
```
ggplot(data_2015_2016, aes(x = factor(fyear), y = ROA, fill = factor(fyear))) +
  geom_violin(scale = "width", adjust = 1) +
  labs(title = "Differences in ROA between 2015 and 2016", x = "Year", y = "ROA") +
  theme_bw() +
  theme(legend.position = "none")
```



The violin plot above shows the difference in *ROA* between 2015 and 2016. This graph shows what we saw earlier in the two descriptive statistics tables, the distribution is very similiar for both years, it differs most significantly at the tails.

5.1.2 Density plot of PE in 2015 & 2016

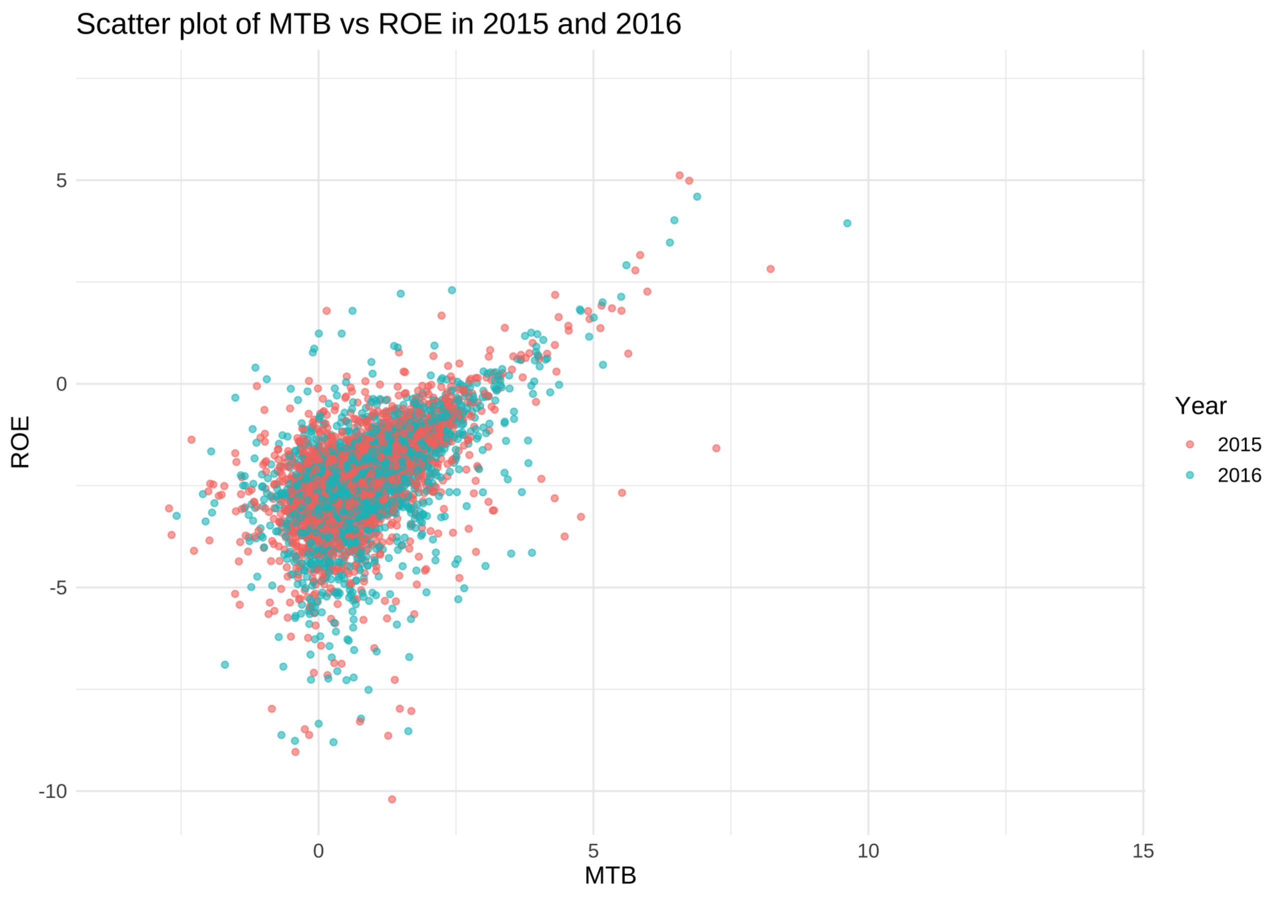
```
ggplot(data_2015_2016, aes(x = PE, fill = factor(fyear))) +
  geom_density(alpha = 0.5) +
  labs(title = "Density plot of PE in 2015 and 2016", x = "PE", y = "Density", fill = "Year") +
  theme_minimal() +
  theme(text = element_text(size = 14))
```



The plot above shows the density of *PE* for the years 2015 and 2016. Just like for *ROE*, the density is very similar between the two years. *PE* is slightly more left-skewed in 2016, however this difference is minimal.

5.1.3 Scatter plot of MTB vs ROE in 2015 & 2016

```
 # Scatter plot for MTB vs ROE
ggplot(data_2015_2016, aes(x = MTB, y = ROE, color = factor(fyear))) +
  geom_point(alpha = 0.6) +
  labs(title = "Scatter plot of MTB vs ROE in 2015 and 2016", x = "MTB", y = "ROE", color = "Year'
  theme_minimal() +
  theme(text = element_text(size = 14))
```



The above scatter plot visualizes the relationship between *MTB* and *ROE* for the years 2015 and 2016. Similarly to the prior visualizations, the data only differs slightly between fiscal years.

5.2 Exploratory data analysis conclusion

In the exploratory data analysis conducted above, we could observe some interesting relationships. Firstly, the data is very similarly distributed for all variables across the two fiscal years. This is consistent with common sense as financials do not usually vary significantly between two years except in crisis, especially considering we removed small firms from the dataset, whose financials would be more volatile. Next, the data also incorporates a significant number of outliers. During EDA we applied logarithm transformation to address these extreme values. However it must be noted that because logarithm transformation reduces the skewness and stabilizes the variance in the data, the similar distributions between the years might also partly stem from the transformation rather than the actual data. More analysis would be needed to definitely conclude this, however this would go beyond the scope of this assignment. Moreover, we found that our predictive models perform better on winsorized data, therefore winsorization will be conducted below on the non log-transformed data for model construction.

6. Model building

After pre-processing the dataset at hand and analyzing the intricate relationships in our data using descriptive statistics, it is now time to construct our predictive models.

6.1 Data preparation

As seen in the tables above, we have some outliers present in our variables of interest. To reduce their effect on the predictive power of the models to be created, we opted to winsorize them with a trim level of 0.05. This means that the lower 5% of the data will be replaced by the 5th percent quantile and the upper 5% of data by the 95th percent quantile. 0.05 is commonly used as a trim level as it provides a good compromise between reducing the effect of extreme outliers in the dataset while retaining as much information as possible. We also winsorized the testing set to keep everything uniform. Before we can construct the models, we need to create a new variable called *PE_smoothed* which represents a PE-ratio with 3-year averaged earnings, as this provides a more robust estimate of a firm's performance than one year's PE-ratio. As it is generally considered bad practice to winsorize the target variable, we opted to perform log-transformation on *PE_smoothed* instead. Additionally, we will only include observations with a positive smoothed PE-ratio. Furthermore, as per the assignment instructions, the models will only be trained on data from 2009. The formula for the smoothed PE-ratio can be seen below:

$$PE_it = \frac{Price_it}{EPS_it + EPS_{i,t-1} + EPS_{i,t-2}}$$

```
 #Selecting variables to winsorize
variables_to_winsorize <- c("ROA", "inv", "size")


trim_level <- 0.05
#Winsorize function
winsorize <- function(x, trim_level) {
  q_lower <- quantile(x, trim_level, na.rm = TRUE)
  q_upper <- quantile(x, 1-trim_level, na.rm = TRUE)
  ifelse(x < q_lower, q_lower, ifelse(x > q_upper, q_upper, x))
}
#Looping over all variables of interest and attaching the winsorized versions as new variables wit
for(var in variables_to_winsorize){
  data[paste0(var,".w")] <- winsorize(data[[var]], trim_level)
  data_testing[paste0(var,".w")] <- winsorize(data_testing[[var]], trim_level)
}



# Group by 'gvkey' and calculate the lagged EPS
data <- data %>%
  arrange(gvkey, fyear) %>%
  group_by(gvkey) %>%
  mutate(lag_EPS1 = lag(EPS, 1), # EPS for t-1
         lag_EPS2 = lag(EPS, 2), # EPS for t-2
         PE_smoothed = 3*price / (EPS + lag_EPS1 + lag_EPS2))

#Filtering model data
data_model <- data %>%
  filter(fyear == 2009, PE_smoothed > 0)

#Setting up cross-validation object with 5 folds
control <- trainControl(method = "cv", number = 5)
```

6.2 Random forest

The ‘Random Forest’ algorithm, a sophisticated ensemble learning method, employs a legion of decision trees to solve complex regression problems. Let’s begin by unpacking the concept of regression. In the grand tapestry of statistical analysis, regression models provide us with a means of predicting a continuous outcome variable from one or more predictor variables. But, when dealing with high-dimensional data, simple linear regression can stumble, failing to capture complex relationships. Enter random forests, an algorithm that doesn’t just build a single tree, but creates an entire forest of them to deliver more accurate predictions. Now, imagine a tree - a decision tree. Each branch represents a decision based on a feature, and the leaf nodes represent the outcome. To create a random forest, we grow many such trees, each subtly different, yet all working towards the same goal. Each tree is trained on a random subset of the data, and at each node, a random subset of features is considered for splitting. This randomness imparts the ‘forest’ with an ability to capture complex patterns and relationships in the data, aiding in regression tasks. But, how does the forest make predictions? Once grown, each tree in the forest makes its own prediction. The final prediction is then calculated as the average of these individual predictions, thus reducing the variance and improving the accuracy of the model. Enter stage: the Out-Of-Bag (OOB) error rate. In the world of random forests, this term refers to a method of measuring the prediction error of the model. While creating the random subsets of data for each tree, some observations may be left out or ‘out-of-bag’. These observations can serve as a validation set. For each observation, the algorithm computes the prediction from only those trees that didn’t have this observation in their training set. The OOB error is then calculated as the average prediction error on these out-of-bag observations. This serves as an internal error estimation, eliminating the need for a separate validation set and providing a robust measure of model performance. To better estimate the performance of the model, cross-validation will also be employed.

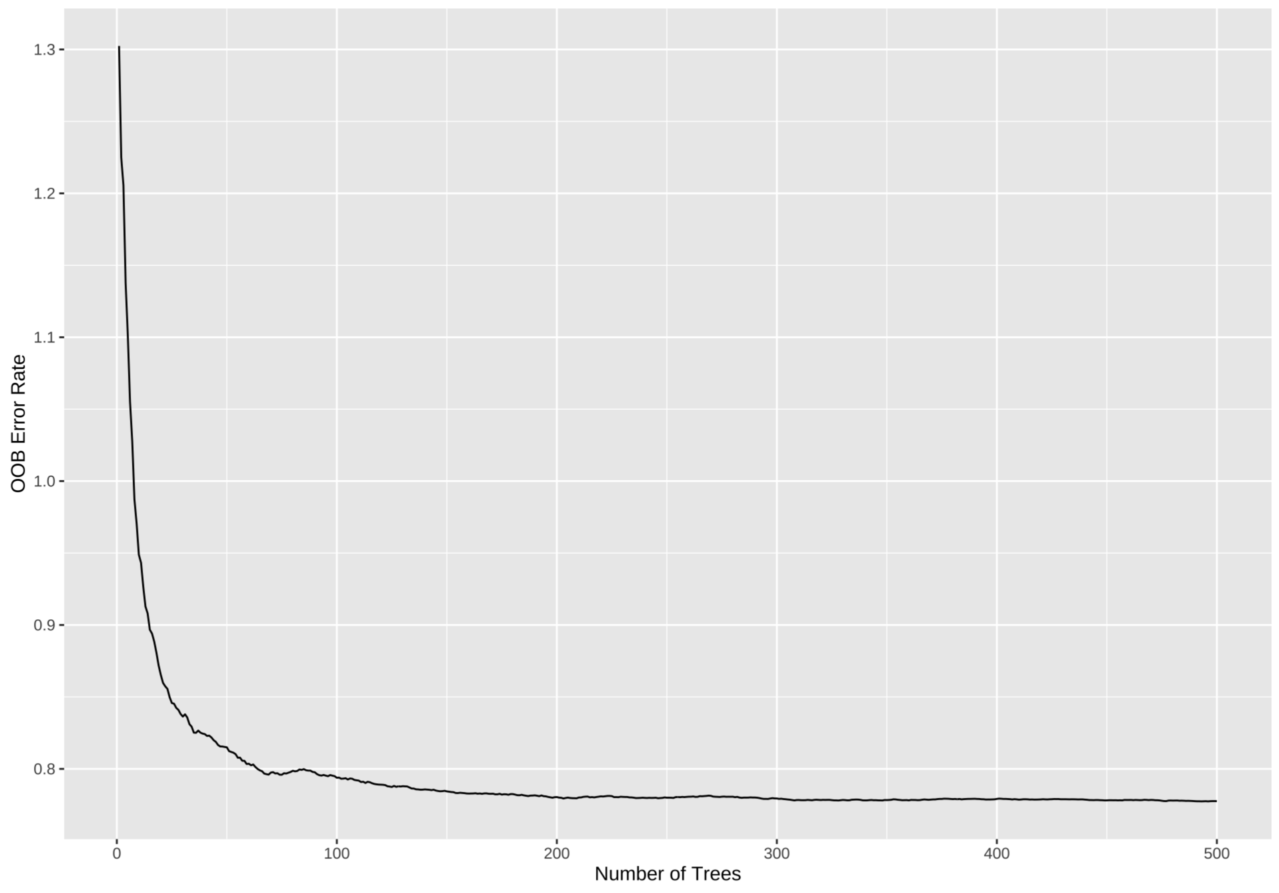
```
 #Training random forest model
forest <- train(log(PE_smoothed) ~ ROA.w + size.w + inv.w, data=data_model, method="rf",
                trControl = control,
                importance = TRUE)
```

```
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
#Showing final model results
print(forest)
```

```
## Random Forest
##
## 3013 samples
##    3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2410, 2410, 2410, 2413, 2409
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared    MAE
##   2     0.8844742  0.02933401  0.6145441
##   3     0.8909379  0.02742711  0.6179918
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
#Plot OOB-error rate
plot(gg_error(forest$finalModel))
```



As can be seen above, the resulting random forest model yields an R-squared value of 0.02933401, meaning that 2.93% of the variance in the data can be explained by our predictor variables. While this R-squared metric might seem small, it is important to consider that a firms PE-ratio is influenced by a multitude of factors that are not included in this model. Moreover, the model yielded a Root mean squared error of 0.8844742. The model also produced a mean absolute error of 0.6145441, which represents the average of the absolute difference between predicted and actual values. However, as we used log transformation for our dependent variable both values does not refer to the absolute error in terms of *PE_smoothed* but rather the logarithm of it. These metrics were computed using 5-fold cross-validation ensuring a more robust estimate of the model's performance. Furthermore, the graph above shows the OOB-error rate vs the number of trees in our random forest model. The error rate decreases with every additional tree in the model, highlighting the power of random forest algorithms.

6.3 Neural network

As we venture further into the complex landscape of artificial intelligence, one marvel of modern computation technology stands tall - Neural Networks. Their uncanny resemblance to the structure of the human brain has garnered much attention. However, these networks don't stop at mere anatomical imitation; they mimic the function of our neurons too, providing a novel approach to problem-solving in the realm of machine learning. Unravel the anatomy of a neural network, and you'll find layers upon layers of interconnected 'neurons', or nodes. Much like the human brain, information flows from one layer to the next, starting from the input layer, progressing through hidden layers, and culminating at the output layer. Each node, a miniature computation unit, takes in input, applies a set of weights, adds a bias, and then runs it through an activation function to determine the output. Now, envision these neural networks as a versatile troupe of actors, taking on roles across a multitude of tasks. They excel at both regression and classification problems, capture intricate patterns in data, and even venture into the uncharted territories of unsupervised learning and reinforcement learning. From recognizing handwritten digits and speech, to driving autonomous vehicles and translating languages, neural networks are transforming the world as we know it. To better estimate the performance of the model, cross-validation will also be employed. The results of the training process can be seen below.

```
 # Train the model using train()
model_nn <- train(log(PE_smoothed) ~ ROA.w + size.w + inv.w,
                  data = data_model,
                  method = "nnet",
                  linout = TRUE,  # linear output for regression
                  trace = FALSE,  # to suppress training messages
                  tuneGrid = data.frame(size = 5, decay = 3),
                  trControl = control)
# Print the model summary
print(model_nn)
```

```
 ## Neural Network
##
## 3013 samples
##    3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2409, 2409, 2411, 2412, 2411
## Resampling results:
##
##   RMSE       Rsquared    MAE
##   0.8540652  0.04783923  0.5863718
##
## Tuning parameter 'size' was held constant at a value of 5
## Tuning
##  parameter 'decay' was held constant at a value of 3
```

In our neural network model we set the size parameter to 5, meaning that there are 5 neurons in each hidden layer. We also opted to set decay to 3, which is a regularization parameter that prevents overfitting by adding a penalty to the loss function based on the size of the weights. Larger weights result in a larger penalty because they make the output of the model more sensitive to changes in the input As can be seen above, the resulting neural network model yields an R-squared value of 0.04429535, meaning that 4.43% of the variance in the data can be explained by our predictor variables. While this R-squared metric might seem small, it is important to consider that a firms PE-ratio is influenced by a multitude of factors that are not included in this model. Moreover, the model yielded a Root mean squared error of 0.8533904. The model also produced a mean absolute error of 0.590711, which represents the average of the absolute difference between predicted and actual values. However, as we used log transformation for our dependent variable both values does not refer to the absolute error in terms of *PE_smoothed* but rather the logarithm of it. These metrics were computed using 5-fold cross-validation ensuring a more robust estimate of the model's performance.

6.4 *PE_smoothed* predictions

In the last step we constructed two powerful predictive models, a random forest and a neural network. We decided to use the neural network to predict *PE_smoothed* on the testing dataset, as it yielded a slightly smaller RMSE and MAE during 5-fold cross-validation, which indicates slightly better accuracy compared to the random forest model. As stated previously, we are using the logarithm of the dependent variable to reduce the effect of outliers. Therefore, we will take the exponent of the predictions to get the actual *PE_smoothed*. These predictions will be saved to a .csv file for later comparison. The code can be seen below

```
 predictions_nn <- exp(predict(model_nn, newdata = data_testing))
write.csv(predictions_nn, file = "/Users/cornelius/Documents/Maastricht University/Business Analyt
```

7. Conclusion

In the above analysis, we constructed sophisticated predictive models that are aimed at accurately predicting a smoothed PE ratio for a large number of firms. The PE ratio is an important metric to predict market changes, assess potential investments, and steer clear of potential financial pitfalls. Before we could construct any models though, we had to clean our dataset. This involved removing missing values and certain observations that would have skewed our analysis results. One example of this would be firms whose total assets are smaller than 0. Afterwards, we conducted an exploratory data analysis for data in the years 2015 and 2016 and investigated the relationships between variables with various plots. Next, we had to perform some further data pre-processing before the models could finally be built. This mainly included winsorizing our independent variables. Afterwards, we constructed to machine learning models, a random forest and a neural network. Ultimatley, we choose the neural network to predict *PE_smoothed*, as it performed slightly better during cross-validation. The resulting predictions have been saved to a .csv file for later comparison. By leveraging the power of data science and predictive modelling, we've seen firsthand how they are reshaping the financial landscape – reducing reliance on human intuition, mitigating the risk of human error, and improving the precision of our forecasting capabilities. This paradigm shift, from intuition-based decisions to data-driven insights, has unlocked a new era in the world of finance. In this assignment, we took strides into this brave new world, beginning to fathom how data science, machine learning, and predictive models, especially concerning PE ratios, decode the complex world of financial markets. As we continue to delve deeper into this rich tapestry, we can expect to uncover even more intriguing insights, equipping us with the knowledge to navigate the dynamic dance of the financial markets with greater confidence and foresight.