Dow Jones Case Study

Executive Summary

In our case study, we present a comprehensive analysis of stock price prediction and risk evaluation using machine learning techniques and financial models. The primary objective was to predict the percentage change in stock prices for the following week, using linear regression, support vector regression (SVR), and decision tree models. Additionally, the Capital Asset Pricing Model (CAPM) was utilized to assess the risk of individual stocks relative to the market.

This study found that SVR outperformed linear regression and decision tree models in terms of predictive accuracy, as shown by lower root mean squared error (RMSE) values. This highlights the effectiveness of SVR in capturing nonlinear relationships between features and stock prices. The application of CAPM provided valuable insights into the risk return tradeoff for individual stocks, helping investors in constructing portfolios.

Recommendations include exploring ensemble learning methods such as random forests or gradient boosting to further enhance predictive performance. Incorporating additional features or external data sources could provide a more comprehensive understanding of market dynamics. Sensitivity analysis is also recommended to assess the durability or model predictions under different scenarios and market conditions.

Overall, this study contributes to the ongoing efforts to improve stock price prediction and risk assessment. By using these recommendations, investors can make more informed decisions, mitigating risks and maximizing returns, improving their chances of success in the world of finance.

Problem

The task we were given for this case study was to predict stock prices and evaluate the risks of stocks. We will build models to determine which stocks will produce the greatest rate of return in the following week. With the goal of maximizing *percent_change_next_weeks_price*, we will explore linear models, decision trees, and support vector regression (SVR) models. Attributes such as high, low, volume, and various pricing metrics will be used in our modeling endeavors. We will assess the accuracy of our models by comparing the root mean squared errors. Additionally, we will evaluate the risks of stocks using the capital asset pricing model

(CAPM). Ultimately, the stock market is difficult to predict, yet we will aim to identify the most appropriate model and discuss our findings. The remainder of this report will include a review of related literature, insights into methodology used, details regarding data preparation and manipulation, and an analysis of our models and findings.

Review of Related Literature

Considering the lucrative benefits of successfully predicting stock prices, it is no surprise that numerous financial analysts and data scientists have attempted to develop a way to accurately predict movements in the stock market. The specific variables used as responses vary among studies, but similar for all is the objective to forecast future stock prices (or index values, returns, etc.) with precision, thereby providing accurate insights for investment.

A previous study by Medarhri et al. (2022)[1] sought to predict the future closing price for five companies in the S&P 500 index using six different machine-learning techniques. Two of these techniques included support vector regression and decision trees, which will be described further in the *Methodology* section. The remaining machine learning techniques employed are k-nearest neighbors (KNN), random forest, multilayer perceptron (MLP), and long short-term memory (LSTM).

As described in the study, KNN is an instance-based learning technique that is easy to build and interpret (Medarhri et al., 2022). It works by finding the *K* closest data points in the training set to a given input and making predictions based on the most common class (for classification) or the average (for regression) of those neighbors. Next, random forest is a more sophisticated machine-learning technique that constructs multiple decision trees during training. It operates by generating random subsets of the training data and features, enabling each decision tree to learn independently and collectively to produce more accurate predictions. The latter two techniques, MLP and LSTM, are types of artificial neural networks. As the name suggests, the multilayer perceptron is a network with an input layer, one or more hidden layers, and an output layer. Each layer contains neurons that connect to neurons in adjacent layers through weighted connections. The MLP employed in this study used a feed-forward algorithm, where information flows in a single direction from input to output. On the other hand, LSTM is a type of recurrent

[1] https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/9934252. Accessed 3/30/2024

neural network. Recurrent connections allow the network to maintain and propagate information over time by feeding the output of a previous time step as input to the next time step.

Different from our study, Medarhri et al. used 10 years of historical data and evaluated the six techniques for each company using two metrics: Mean absolute error (MAE) and root mean squared error (RMSE). The study found that the LSTM and MLP had the overall best performance considering both performance metrics. However, the authors concluded that there was no best model across all datasets (i.e. companies), "since the optimal configuration of a given ML depends mainly on the characteristic of the dataset" (Medarhri et al., 2022). In other words, the best machine learning model for predicting stock prices varied depending on the unique features of each stock.

Methodology

As described in the *Problem* section, the goal of this case study is to predict the percentage change in stock prices of the following week (*percent_change_next_week_price*), a continuous variable, making this case study a regression task. Each record in the provided dataset contains data for a week, which is reflected in the *date* variable. Another variable, *quarter*, will be used to split the data into a training set (quarter 1) and a testing set (quarter 2). The identifier variable, *stock*, indicates the ticker symbol for the given company. The remaining variables in the provided dataset are continuous predictors such as opening price, closing price, highest and lowest price of the stock during the week, etc. Further details on how and why variables were dropped, cleaned, and selected for modeling will be provided in the *Data* and *Results* sections.

In this analysis, we will test and compare three different methods to predict the percentage change in stock prices. First, we will use linear regression, a simple machine-learning algorithm, that fits the best linear line between the response variable and one or more predictors. Linear regression allows for high interpretability by providing coefficients for each predictor, but the method is limited by relying on several assumptions, such as linearity, homoscedasticity, and no multicollinearity. Because multicollinearity can lead to unstable estimates of the regression coefficients, we will examine the degree of correlation among predictors to ensure the validity of this assumption.

Secondly, we will employ support vector regression (SVR), a technique that seeks to find the hyperplane that best fits the training data while minimizing prediction error. Where linear

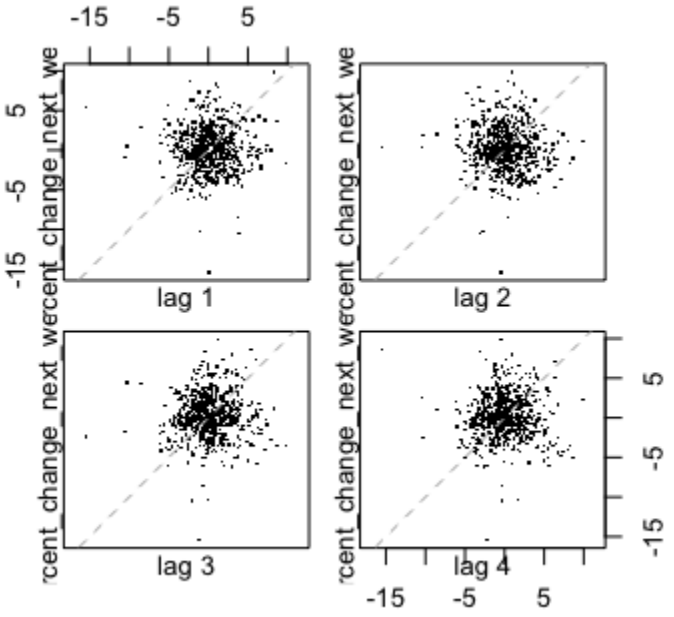
regression aims to minimize the error between actual and predicted values, SVR aims to minimize the *margin* between the predicted values and the actual values within a certain tolerance, controlled by tuning parameters such as gamma and C. This makes SVR particularly useful for regression tasks where the data may be noisy or contain outliers. A potential limitation of SVR, however, is that it typically requires more data to train effectively, especially when dealing with high-dimensional feature spaces. Also different from linear regression, SVR is a non-parametric technique, meaning it has no assumptions about the distribution of the data. Additionally, SVR can handle non-linear relationships by using kernel functions. In this analysis, we will utilize the radial basis function (RBF) to capture nonlinear relationships between our response and predictor variables. We will tune the model by selecting the best gamma from the range of .01 and .1 and the best value of C from the range of .1 to 1 to avoid overfitting on the training data.

Finally, we will predict changes in stock price using a decision tree. Decision trees offer an intuitive approach to modeling complex relationships by repeatedly breaking down the data into smaller groups based on their characteristics. They are capable of handling both numerical and categorical data and are easily interpretable. However, decision trees are prone to overfitting, especially when the tree depth is not properly controlled or when the dataset contains noise or outliers. Regularization techniques such as pruning and limiting the maximum depth of the tree can limit overfitting. This method concludes our modeling for predicting stock prices. We will evaluate and compare models on accuracy by calculating the mean root squared error (MRSE).

To address the second objective of this case study, we will use the capital asset pricing model (CAPM) to assess the risk of different stocks. CAPM calculates the expected return on a stock based on its systemic risk, measured by its beta coefficient, which indicates the stock's sensitivity to market movements. A beta greater than 1 indicates a stock is riskier than the market, while lower betas suggest that a stock will reduce the risk of a portfolio. CAPM does not explicitly consider *un*systematic risk, but it assumes investors hold well-diversified portfolios to account for this type of risk. CAPM also relies on other assumptions that may not always hold true, such as no transaction costs and equally informed investors.

Data

The dataset provided for this project contains weekly stock data between 1/7/2011 and 6/3/2011. After removing rows containing missing values in the two variables with numbers from the previous week (i.e. the first observation for each stock), the data contains 720 observations, 24 for each stock starting from 1/14/2011. Before splitting our data into train and test, we had a few clean-up tasks. First, we needed to remove dollar signs from the columns *open*, *high*, *low*, and *close*. We then converted these columns from the character data type to numeric. Next, we created a column called *week*, where we numbered each week 1-24 for each unique stock. Finally, we removed the columns *next_weeks_open*, *next_weeks_close*, and *date*. We knew we needed to remove *next_weeks_open* and *next_weeks_close* because we could not use data related to the following week when trying to predict the outcomes for that week. We also removed the *date* column because it was unnecessary for modeling, and the *week* variable that we created serves as a marker of time. Before continuing, we wanted to see if there was any impact on using lag on the response variable, *percent_change_next_weeks_price*. As seen in the lag plot, lagging the response variable does not appear to have an impact. Therefore, we will not lag the response variable.



(Figure 1.)

We ended our data manipulation by splitting it into train and test sets. We used quarter 1 (Jan-Mar) data for training and quarter 2 (Apr-Jun) data for testing. For our CAPM analysis, we extracted S&P 500 data from Yahoo Finance for the time period corresponding with our original

dataset. The data included *Date*, *Open*, *High*, *Low*, *Close*, *Adj Close*, and *Volume*. We also created a new variable, *Week*, to number each week 1-24.

Results

Before fitting the linear model, we first checked for multicollinearity among predictor variables using Variance Inflation Factor (VIF) analysis. According to the VIF results, we removed the variables with high VIF values (>10) and our final predictors for the models are: *low*, *volume*, *percent_change_price*, *percent_change_volume_over_last_wk*, *days_to_next_dividend*, *percent_return_next_dividend*, and *previous_weeks_volume*. We used the same set of predictors for the SVR and decision tree models. While these methods are generally insensitive to multicollinearity, the decision to exclude highly correlated variables was based on the understanding that their inclusion would not theoretically enhance performance, given their strong correlation with variables retained in the models.

We then fitted linear regression models, SVR models, and decision tree models for each stock using the training set and predicting on the testing set. The performance of each model was evaluated on the RMSE metric, and the results obtained from the analysis are summarized below:

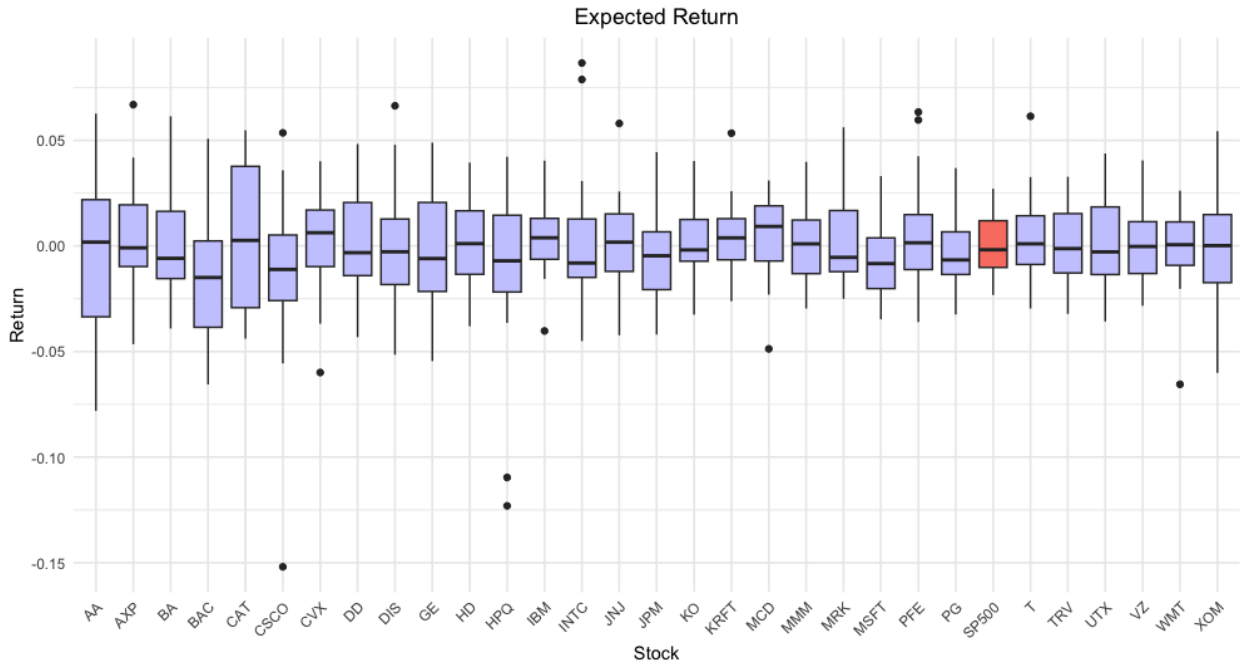
Linear Regression: 9.984327

SVR: 2.880562

Decision Tree: 3.151962

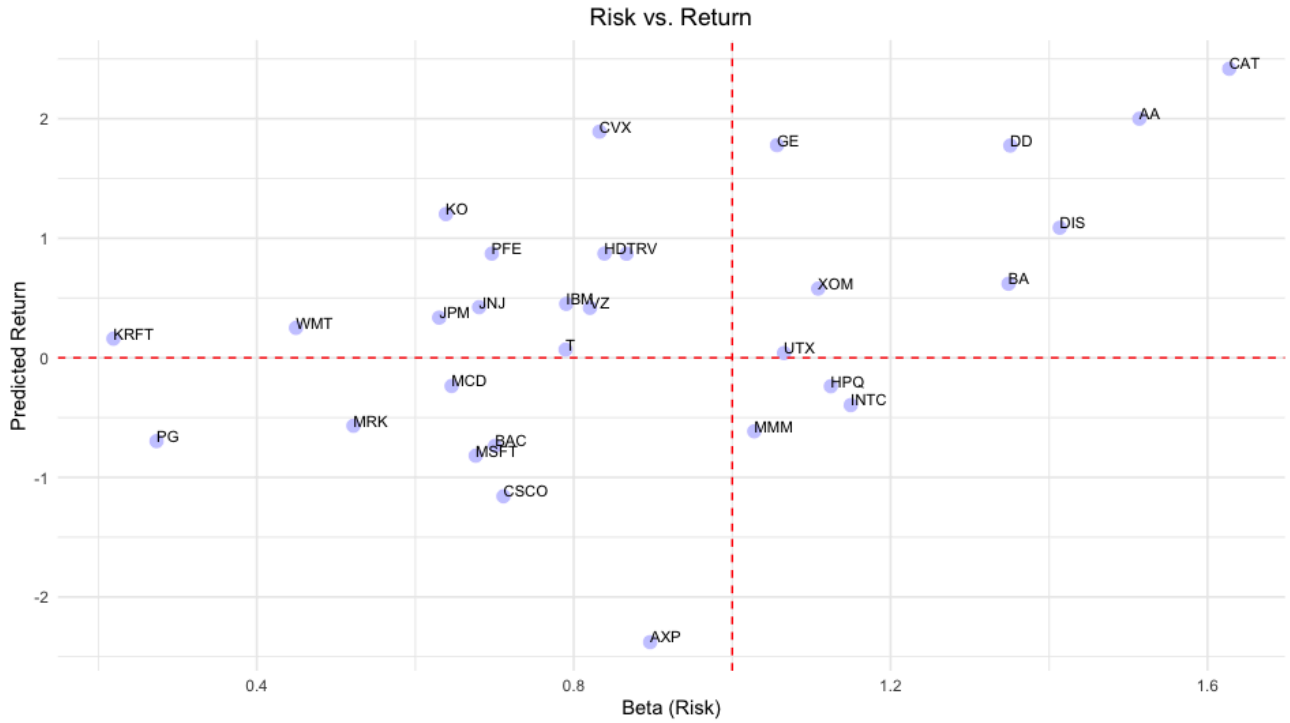
In conclusion, SVR outperformed linear regression and decision tree models in terms of predictive accuracy, as evidenced by the lower RMSE values, suggesting that it captures the nonlinear relationships between features and stock prices more effectively.

To determine risk for different stocks, we applied the Capital Asset Pricing Model (CAPM) on the Dow Jones data and S&P 500 data sourced from Yahoo Finance by calculating returns and then estimating betas for individual stocks. The expected returns and standard deviations of the S&P 500 index and individual stocks are computed. Beta coefficients, representing the sensitivity of individual stocks to market movements (S&P 500), are calculated using linear regression models. As mentioned in the *Methodology* section, stocks with beta coefficients greater than 1 are considered riskier than the market, while those with beta coefficients less than 1 are considered less risky.



(Figure 2. expected returns of individual stocks compared to the S&P 500)

The CAPM results indicated several key findings. Firstly, the boxplot visualization above illustrates that the volatility of stocks is generally lower for the S&P 500 index compared to individual stocks in the Dow Jones. This aligns with the foundational principle of CAPM, which suggests that diversified portfolios, such as the S&P 500, tend to exhibit lower risk due to their exposure to a broader range of assets. Additionally, the estimation of beta coefficients (see appendix) reveals varying levels of systematic risk for individual stocks relative to the market represented by the S&P 500. Stocks with betas greater than 1, such as Alcoa Corp (AA) and Caterpillar (CAT), are considered more volatile than the market, while those with betas less than 1, like Johnson & Johnson (JNJ) and Kraft Foods Group (KRFT), are less volatile. These patterns are also reflected in the variance of the boxplots above.



(Figure 3. scatter plot for Risk vs. Return for each stock)

Using the trained SVR to obtain predictions for week 14, we can plot the predicted returns against the betas for each stock as seen above in Figure 3. To enhance analysis, the plot is partitioned into four quadrants utilizing beta=1 and return=0 as dividers: Quadrant 1 signifies stocks with high returns and low risks, Quadrant 2 represents stocks with high returns and high risks, Quadrant 3 denotes stocks with low returns and low risks, and Quadrant 4 reflects stocks with low returns and high risks.

By categorizing stocks into four quadrants based on their beta and return, investors can quickly identify opportunities for diversification and risk management. For instance, stocks positioned in Quadrant 1, like Coca-Cola Company (KO) and Chevron Corporation (CVX), offer potentially high returns with low risks, appealing to conservative investors. Conversely, stocks in Quadrant 2, such as Caterpillar Inc. (CAT) and Alcoa Corporation (AA), present higher risks but also higher return potential, suitable for more aggressive investment strategies. This analysis allows investors to adjust their portfolios strategically, balancing risk and return to align with their financial goals and risk tolerance.

Conclusion & Recommendations

In conclusion, our case study aimed to predict stock prices and evaluate stock risks using various machine-learning techniques and financial models. Through a comprehensive analysis, it was found that support vector regression (SVR) outperformed linear regression and decision tree models in terms of predictive accuracy, with lower root mean squared error (RMSE) values. This suggests that SVR effectively captures nonlinear relationships between features and stock prices. Additionally, the Capital Asset Pricing Model (CAPM) provided valuable insights into the risk-return tradeoff for individual stocks relative to the market. By leveraging predictions, investors can make informed decisions about buying, selling, or holding stocks.

Moving forward, it is recommended to explore ensemble learning methods such as random forests or gradient boosting to further improve predictive performance. Incorporating additional features or external data sources could also enhance the models' ability to capture comprehensive information about market dynamics. Conducting sensitivity analysis would be beneficial to assess the robustness of model predictions under different scenarios and market conditions.

By implementing these recommendations, investors can make more informed decisions and construct well-diversified portfolios tailored to their risk preferences and investment objectives. Overall, this case study contributes to the ongoing efforts to improve stock price prediction and risk assessment, providing valuable insights for researchers in the field of finance and data science.

Appendix

```
##### Train/Test Split

# Splitting into train (quarter 1) and test (quarter 2)
train <- data %>%
  filter(quarter == 1) %>%
  select(-quarter)
test <- data %>%
```

```
 filter(quarter == 2) %>%
 select(-quarter)



##### Modeling

### Linear Model:

# Checking for multicollinearity
vif_check <- lm(percent_change_next_weeks_price ~ low + volume +
          percent_change_price + percent_change_volume_over_last_wk +
          days_to_next_dividend + percent_return_next_dividend +
           previous_weeks_volume, data = train)
vif(vif_check) # Removed variables: close, high, open

##                       low                    volume
##                  1.599743                  7.241811
##      percent_change_price percent_change_volume_over_last_wk
##                  1.207362                  1.811323
##      days_to_next_dividend      percent_return_next_dividend
##                  1.020365                  1.256904
##      previous_weeks_volume
##                  6.803753

# Getting unique stock values
unique_stocks <- as.factor(unique(train$stock))

# Creating data frame for performance metrics (RMSE, R-Sq, )
RMSE = rep(NA, length(unique_stocks))
RSq = rep(NA, length(unique_stocks))
lm.results = data.frame("Stock" = unique_stocks, "RMSE" = RMSE, "R-Squared" = RSq)

# Looping through each stock in the train set
for(i in 1:length(unique_stocks)) {
 # Subsetting train and test data set for current stock
```

```
 subset_train <- train %>%
        subset(stock == unique_stocks[i])
 subset_test <- test %>%
        subset(stock == unique_stocks[i])

 # Fitting the linear regression model using subset data
 lm.fit <- lm(percent_change_next_weeks_price ~ low + volume +
                percent_change_price + percent_change_volume_over_last_wk +
                days_to_next_dividend + percent_return_next_dividend +
                previous_weeks_volume, data = subset_train)

 lm.preds = predict(lm.fit, subset_test)

 # Calculating RMSE
 lm.rmse <- rmse(subset_test$percent_change_next_weeks_price, lm.preds)
 lm.results[i, "RMSE"] <- lm.rmse

 # Calculating R-squared
 lm_r_squared <- summary(lm.fit)$r.squared
 lm.results[i, "R_squared"] <- lm_r_squared



### SVR:

svr.results = data.frame("Stock" = unique_stocks, "RMSE" = RMSE)

for(i in 1:length(unique_stocks)) {
 subset_train <- train %>%
        subset(stock == unique_stocks[i])
 subset_test <- test %>%
        subset(stock == unique_stocks[i])

 set.seed(123) ; tuned = tune.svm(percent_change_next_weeks_price ~ low + volume +
```

```
                        percent_change_price + percent_change_volume_over_last_wk +
                        days_to_next_dividend + percent_return_next_dividend +
                        previous_weeks_volume, data = subset_train,
                        gamma = seq(.01, .1, by = .01),
                        cost = seq(.1, 1, by = .1), scale = TRUE)

 svr.fit <- svm(percent_change_next_weeks_price ~ low + volume +
                  percent_change_price + percent_change_volume_over_last_wk +
                  days_to_next_dividend + percent_return_next_dividend +
                  previous_weeks_volume, data = subset_train,
          gamma = tuned$best.parameters$gamma, cost = tuned$best.parameters$cost)

 svr.preds = predict(svr.fit, subset_test, type = "response")

 svr.rmse <- sqrt(mean((svr.preds - subset_test$percent_change_next_weeks_price)^2))
 svr.results[i, "RMSE"] <- svr.rmse


### Decision Tree:

tree.results = data.frame("Stock" = unique_stocks, "RMSE" = RMSE)

for(i in 1:length(unique_stocks)) {
 subset_train <- train %>%
          subset(stock == unique_stocks[i])
 subset_test <- test %>%
          subset(stock == unique_stocks[i])

 tree.fit <- tree(percent_change_next_weeks_price ~ low + volume +
                  percent_change_price + percent_change_volume_over_last_wk +
                  days_to_next_dividend + percent_return_next_dividend +
                  previous_weeks_volume, data = subset_train)

 tree.preds = predict(tree.fit, subset_test)
```

```
tree.rmse <- rmse(subset_test$percent_change_next_weeks_price, tree.preds)
tree.results[i, "RMSE"] <- tree.rmse


##### Comparing models
mean(lm.results$RMSE)

## [1] 9.984327

mean(svr.results$RMSE)

## [1] 2.880562

mean(tree.results$RMSE)

## [1] 3.151962

# CAPM: Get S&P500 data from yahoo.finance
SP <- read_excel("SP500.xlsx")
SP <- SP %>% arrange(Date)
week = seq(from = 1, to = dim(SP)[1], by = 1)
SP <- cbind(week, SP)

# Computing the returns on the close price for SP500 and each stock in Dow Jones
ReturnSP = Delt(SP[,6])

ReturnAA <- data %>% filter(stock == "AA") %>% pull(close) %>% Delt()
ReturnAXP <- data %>% filter(stock == "AXP") %>% pull(close) %>% Delt()
ReturnBA <- data %>% filter(stock == "BA") %>% pull(close) %>% Delt()
ReturnBAC <- data %>% filter(stock == "BAC") %>% pull(close) %>% Delt()
ReturnCAT <- data %>% filter(stock == "CAT") %>% pull(close) %>% Delt()
ReturnCSCO <- data %>% filter(stock == "CSCO") %>% pull(close) %>% Delt()
ReturnCVX <- data %>% filter(stock == "CVX") %>% pull(close) %>% Delt()
ReturnDD <- data %>% filter(stock == "DD") %>% pull(close) %>% Delt()
ReturnDIS <- data %>% filter(stock == "DIS") %>% pull(close) %>% Delt()
ReturnGE <- data %>% filter(stock == "GE") %>% pull(close) %>% Delt()
```

```
ReturnHD <- data %>% filter(stock == "HD") %>% pull(close) %>% Delt()
ReturnHPQ <- data %>% filter(stock == "HPQ") %>% pull(close) %>% Delt()
ReturnIBM <- data %>% filter(stock == "IBM") %>% pull(close) %>% Delt()
ReturnINTC <- data %>% filter(stock == "INTC") %>% pull(close) %>% Delt()
ReturnJNJ <- data %>% filter(stock == "JNJ") %>% pull(close) %>% Delt()
ReturnJPM <- data %>% filter(stock == "JPM") %>% pull(close) %>% Delt()
ReturnKO <- data %>% filter(stock == "KO") %>% pull(close) %>% Delt()
ReturnKRFT <- data %>% filter(stock == "KRFT") %>% pull(close) %>% Delt()
ReturnMCD <- data %>% filter(stock == "MCD") %>% pull(close) %>% Delt()
ReturnMMM <- data %>% filter(stock == "MMM") %>% pull(close) %>% Delt()
ReturnMRK <- data %>% filter(stock == "MRK") %>% pull(close) %>% Delt()
ReturnMSFT <- data %>% filter(stock == "MSFT") %>% pull(close) %>% Delt()
ReturnPFE <- data %>% filter(stock == "PFE") %>% pull(close) %>% Delt()
ReturnPG <- data %>% filter(stock == "PG") %>% pull(close) %>% Delt()
ReturnT <- data %>% filter(stock == "T") %>% pull(close) %>% Delt()
ReturnTRV <- data %>% filter(stock == "TRV") %>% pull(close) %>% Delt()
ReturnUTX <- data %>% filter(stock == "UTX") %>% pull(close) %>% Delt()
ReturnWMT <- data %>% filter(stock == "WMT") %>% pull(close) %>% Delt()
ReturnVZ <- data %>% filter(stock == "VZ") %>% pull(close) %>% Delt()
ReturnXOM <- data %>% filter(stock == "XOM") %>% pull(close) %>% Delt()

# Combining data
All_Returns = cbind(ReturnSP, ReturnAA, ReturnAXP, ReturnBA, ReturnBAC, ReturnCAT,
ReturnCSCO, ReturnCVX, ReturnDD, ReturnDIS, ReturnGE, ReturnHD, ReturnHPQ, ReturnIBM,
ReturnINTC, ReturnJNJ, ReturnJPM, ReturnKO, ReturnKRFT, ReturnMCD, ReturnMMM, ReturnMRK,
ReturnMSFT, ReturnPFE, ReturnPG, ReturnT, ReturnTRV, ReturnUTX, ReturnWMT, ReturnVZ,
ReturnXOM)

colnames(All_Returns) = c("SP500", "AA", "AXP", "BA", "BAC", "CAT", "CSCO", "CVX", "DD",
"DIS", "GE", "HD", "HPQ", "IBM", "INTC", "JNJ", "JPM", "KO", "KRFT", "MCD", "MMM", "MRK",
"MSFT", "PFE", "PG", "T", "TRV", "UTX", "WMT", "VZ", "XOM")

All_Returns <- na.omit(All_Returns)
```

```r
# Creating boxplot to show the volatility of all individual stocks and S&P 500.
all_stocks <- c(unique_stocks, "SP500")

All_Returns_long <- tidyr::gather(data.frame(All_Returns), key = "Stock", value = "Return")

sp500_fill <- "salmon"

boxplot <- ggplot(All_Returns_long, aes(x = Stock, y = Return,
                                        fill = ifelse(Stock == "SP500", "SP500", "Others"))) +
  geom_boxplot() +
  scale_fill_manual(values = c("Others" = "#CCCCFF", "SP500" = sp500_fill)) +
  labs(x = "Stock", y = "Return", title = "Expected Return") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  guides(fill = FALSE)

print(boxplot)

# We see that the risk is much lesser in the S&P compared to the individual stocks.

# Computing mean and standard deviation for the returns.
DataMean = apply(All_Returns, 2, mean)
DataSD = apply(All_Returns, 2, sd)
cbind(DataMean, DataSD)

# Fitting linear models to get betas
lm.AA <-  lm(AA ~ SP500, data = as.data.frame(All_Returns))
BetaAA <- summary(lm.AA)$coefficients[2, 1]

lm.AXP <- lm(AXP ~ SP500, data = as.data.frame(All_Returns))
BetaAXP <- summary(lm.AXP)$coefficients[2, 1]

lm.BA <- lm(BA ~ SP500, data = as.data.frame(All_Returns))
BetaBA <- summary(lm.BA)$coefficients[2, 1]
```

```
lm.BAC <- lm(BAC ~ SP500, data = as.data.frame(All_Returns))
BetaBAC <- summary(lm.BAC)$coefficients[2, 1]

lm.CAT <- lm(CAT ~ SP500, data = as.data.frame(All_Returns))
BetaCAT <- summary(lm.CAT)$coefficients[2, 1]

lm.CSCO <- lm(CSCO ~ SP500, data = as.data.frame(All_Returns))
BetaCSCO <- summary(lm.CSCO)$coefficients[2, 1]

lm.CVX <- lm(CVX ~ SP500, data = as.data.frame(All_Returns))
BetaCVX <- summary(lm.CVX)$coefficients[2, 1]

lm.DD <- lm(DD ~ SP500, data = as.data.frame(All_Returns))
BetaDD <- summary(lm.DD)$coefficients[2, 1]

lm.DIS <- lm(DIS ~ SP500, data = as.data.frame(All_Returns))
BetaDIS <- summary(lm.DIS)$coefficients[2, 1]

lm.GE <- lm(GE ~ SP500, data = as.data.frame(All_Returns))
BetaGE <- summary(lm.GE)$coefficients[2, 1]

lm.HD <- lm(HD ~ SP500, data = as.data.frame(All_Returns))
BetaHD <- summary(lm.HD)$coefficients[2, 1]

lm.HPQ <- lm(HPQ ~ SP500, data = as.data.frame(All_Returns))
BetaHPQ <- summary(lm.HPQ)$coefficients[2, 1]

lm.IBM <- lm(IBM ~ SP500, data = as.data.frame(All_Returns))
BetaIBM <- summary(lm.IBM)$coefficients[2, 1]

lm.INTC <- lm(INTC ~ SP500, data = as.data.frame(All_Returns))
BetaINTC <- summary(lm.INTC)$coefficients[2, 1]
```

```
lm.JNJ <- lm(JNJ ~ SP500, data = as.data.frame(All_Returns))
BetaJNJ <- summary(lm.JNJ)$coefficients[2, 1]

lm.JPM <- lm(JPM ~ SP500, data = as.data.frame(All_Returns))
BetaJPM <- summary(lm.JPM)$coefficients[2, 1]

lm.KO <- lm(KO ~ SP500, data = as.data.frame(All_Returns))
BetaKO <- summary(lm.KO)$coefficients[2, 1]

lm.KRFT <- lm(KRFT ~ SP500, data = as.data.frame(All_Returns))
BetaKRFT <- summary(lm.KRFT)$coefficients[2, 1]

lm.MCD <- lm(MCD ~ SP500, data = as.data.frame(All_Returns))
BetaMCD <- summary(lm.MCD)$coefficients[2, 1]

lm.MMM <- lm(MMM ~ SP500, data = as.data.frame(All_Returns))
BetaMMM <- summary(lm.MMM)$coefficients[2, 1]

lm.MRK <- lm(MRK ~ SP500, data = as.data.frame(All_Returns))
BetaMRK <- summary(lm.MRK)$coefficients[2, 1]

lm.MSFT <- lm(MSFT ~ SP500, data = as.data.frame(All_Returns))
BetaMSFT <- summary(lm.MSFT)$coefficients[2, 1]

lm.PFE <- lm(PFE ~ SP500, data = as.data.frame(All_Returns))
BetaPFE <- summary(lm.PFE)$coefficients[2, 1]

lm.PG <- lm(PG ~ SP500, data = as.data.frame(All_Returns))
BetaPG <- summary(lm.PG)$coefficients[2, 1]

lm.T <- lm(T ~ SP500, data = as.data.frame(All_Returns))
BetaT <- summary(lm.T)$coefficients[2, 1]

lm.TRV <- lm(TRV ~ SP500, data = as.data.frame(All_Returns))
```

```
BetaTRV <- summary(lm.TRV)$coefficients[2, 1]

lm.UTX <- lm(UTX ~ SP500, data = as.data.frame(All_Returns))
BetaUTX <- summary(lm.UTX)$coefficients[2, 1]

lm.WMT <- lm(WMT ~ SP500, data = as.data.frame(All_Returns))
BetaWMT <- summary(lm.WMT)$coefficients[2, 1]

lm.VZ <- lm(VZ ~ SP500, data = as.data.frame(All_Returns))
BetaVZ <- summary(lm.VZ)$coefficients[2, 1]

lm.XOM <- lm(XOM ~ SP500, data = as.data.frame(All_Returns))
BetaXOM <- summary(lm.XOM)$coefficients[2, 1]

# Create a vector of stock names
stock_names <- c("AXP", "BA", "BAC", "CAT", "CSCO", "CVX", "DD", "DIS", "GE", "HD",
                 "HPQ", "IBM", "INTC", "JNJ", "JPM", "KO", "KRFT", "MCD", "MMM",
                 "MRK", "MSFT", "PFE", "PG", "T", "TRV", "UTX", "WMT", "VZ", "XOM")

# Create a vector of corresponding beta coefficients
beta_values <- c(BetaAXP, BetaBA, BetaBAC, BetaCAT, BetaCSCO, BetaCVX, BetaDD, BetaDIS,
BetaGE, BetaHD, BetaHPQ, BetaIBM, BetaINTC, BetaJNJ, BetaJPM, BetaKO, BetaKRFT, BetaMCD,
BetaMMM, BetaMRK, BetaMSFT, BetaPFE, BetaPG, BetaT, BetaTRV, BetaUTX, BetaWMT, BetaVZ,
BetaXOM)

# Create a data frame
beta_df <- data.frame(Stock = stock_names, Beta = beta_values)

# Display the data frame
print(beta_df)

##        Stock    Beta

## 1     AA 1.5139729
## 2      AXP 0.8964122
```

```
## 3      BA 1.3487842
## 4     BAC 0.7006986
## 5     CAT 1.6273609
## 6    CSCO 0.7112372
## 7     CVX 0.8324828
## 8      DD 1.3509593
## 9     DIS 1.4134364
## 10     GE 1.0565306
## 11     HD 0.8387548
## 12    HPQ 1.1244552
## 13    IBM 0.7905043
## 14   INTC 1.1496246
## 15    JNJ 0.6804793
## 16    JPM 0.6300883
## 17     KO 0.6385584
## 18   KRFT 0.2189589
## 19    MCD 0.6457662
## 20    MMM 1.0278486
## 21    MRK 0.5219019
## 22   MSFT 0.6762366
## 23    PFE 0.6965208
## 24     PG 0.2733772
## 25      T 0.7897059
## 26    TRV 0.8666337
## 27    UTX 1.0654118
## 28    WMT 0.4493817
## 29     VZ 0.8204560
## 30    XOM 1.1084921
```

Comparing Risk and Return for Each Stock

```
# Extracting predictions for week 13 for each stock and creating a df for week 14
week_13_predictions <- sapply(predictions, `[`, 13)
```

```
week_14_data <- data.frame(
  Stock = unique_stocks,
  Prediction = week_13_predictions,
  stringsAsFactors = FALSE)

# print(week_14_data)

# Merging beta values with week 14 predictions
merged_data <- merge(beta_df, week_14_data, by = "Stock")

# Plotting risk against return
scatterplot <- ggplot(merged_data, aes(x = Beta, y = Prediction)) +
  geom_point(color = "#CCCCFF", size = 3) +
  geom_text(aes(label = Stock), hjust = 0, vjust = 0, size = 3) +
  labs(x = "Beta (Risk)", y = "Predicted Return", title = "Risk vs. Return") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_vline(xintercept = 1, linetype = "dashed", color = "red") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

print(scatterplot)
```