Connor Allison

Joel Fry

Nicholas Mata

Royalpriesthood Olola

DA-6813

# Dow Jones Case Study

## Executive Summary

This study explores the use of machine learning models to predict stock price changes for the Dow Jones Index. By applying Linear Regression (LM), Support Vector Regression (SVR), and Decision Tree (DT) models, we aim to forecast *percent_change_next_weeks_price* based on *previous_weeks_volume*. The models are compared based on their Mean Squared Error (MSE), with an additional risk analysis performed using the Capital Asset Pricing Model (CAPM). Further analysis was conducted via a Monte Carlo simulation to provide an alternative method for risk assessment. Our analysis finds that the SVR with radial kernel performs best in terms of prediction accuracy, while Linear Regression provides insights into the most significant factors affecting price movements.

## Problem

The challenge is to predict stock price percentage changes for the next week based on weekly performance data. Identifying the best model to forecast price movements with accuracy is crucial for making informed financial decisions. The study compares three machine learning techniques (LM, SVR, and DT) to determine which model provides the most reliable predictions of stock price changes based on historical data, particularly *previous_weeks_volume*. This report will first review related work, followed by a description of the methodology, a discussion of results, and a conclusion on how the findings can be applied practically.

## Literature Review

Stock price prediction has been a subject of interest in financial modeling, with various machine learning methods being applied to predict future price movements (Krollner et al., 2013). Linear regression has been commonly used for predicting price changes,

though its performance in stock market prediction can be limited by non-linear relationships (Kou et al., 2015). Support Vector Regression, particularly with non-linear kernels, has been found to improve prediction accuracy in complex datasets (Wang et al., 2014). Decision Trees, on the other hand, capture non-linear patterns and interactions that may not be easily modeled by traditional methods. The combination of these methods provides a comprehensive approach to understanding stock price dynamics (Bansal, Malti, et al 2022).

## Methodology

We plan to use three different machine learning models to predict *percent_change_next_weeks_price* based on *previous_weeks_volume*. These models will be trained on historical stock data, and their performance will be evaluated using Mean Squared Error (MSE). Additionally, CAPM will be used to assess stock risk and volatility. Linear Regression (LR), Support Vector Regression (SVR), and Decision Trees (DT) are the models that will be used, each with distinct assumptions.

Linear Regression assumes a linear relationship between the independent variables (e.g., previous week's volume) and the target (e.g., stock price change). It also assumes independence of errors and homoscedasticity (constant variance of residuals), as well as a normal distribution of errors for inference. While easy to interpret, LR may not perform well with highly volatile financial data due to its inability to capture non-linear relationships (Abraham and Ledolter 1983). This model will explore the linear relationship between *previous_weeks_volume* and *percent_change_next_weeks_price*. Lag variables as well as a stepwise selection using all variables for the model will be explored to see if any improvements are made.

SVR uses different kernels to model non-linear relationships by mapping data to a higher-dimensional space. Assumptions include linearity in transformed space, error tolerance, allowing the model to ignore small errors, and sparsity of support vectors to define the decision boundary. SVR is effective in capturing complex patterns in stock price data, but choosing the right kernel and parameters can be computationally intensive and sensitive to noise (Smola and Scholkopf 2004). Using a radial kernel, this model will capture non-linear relationships in hopes to improve prediction accuracy.

Decision Trees split data based on feature values, capturing non-linear relationships. Key assumptions are feature independence within each split, no need for feature scaling, risk of overfitting. DTs excel at handling complex interactions but can overfit without pruning or regularization, which is a concern in stock market prediction (Breiman, Leo, et al 2017).

We will compare the models based on their MSE, selecting the model that minimizes this error as the most effective for stock price prediction. To ensure our comparisons are accurate, we will train our models on 1st quarter data and evaluate them on 2nd quarter data for all 30 stocks in the dataset. Once we have the MSE's from all models for each stock, the average MSE for each model over the 30 stocks will be used to conclude the most effective model. We will use the most effective model to compute beta values and along with the CAPM, we will assess the risks of the stocks.

**Data**

The dataset consists of stock data from the Dow Jones Index, with each record representing weekly stock performance. The target variable is *percent_change_next_weeks_price*, and key attributes include:

Market data: *quarter, stock,* and *date*

Price data: *open*, *high*, *low*, *close*, *next_weeks_open* and *next_weeks_close*.

Volume metrics: *volume* and *previous_weeks_volume*.

Percentage change metrics: *percent_change_volume_over_last_wk, percent_change_price* and *percent_change_next_weeks_price*.

Dividend information: *days_to_next_dividend* and *percent_return_next_dividend*.

The dataset was cleaned and transformed to ensure consistency: Price values were converted from string format with currency symbols to numeric values. Percentage changes in price were transformed into numerical types. The *quarter* and *stock* columns were encoded as factors for analysis. The *date* column was parsed into a standard date format. The amount of data between Q1 and Q2 was close to evenly split with 12 observations per stock in Q1 and 13 for Q2. Correlation between numeric variables used in the model were also found to not be highly correlated with each other.

| | |
|---|---|
| `test_stock` | `13 obs. of 16 variables` |
| `train_stock` | `12 obs. of 16 variables` |

## Results

A linear regression model, SVR, and decision tree were trained for each stock to model how *previous_weeks_volume* influences *percent_change_next_weeks_price*. The models were evaluated based on Mean Squared Error (MSE), with the lowest average MSE indicating the best model.

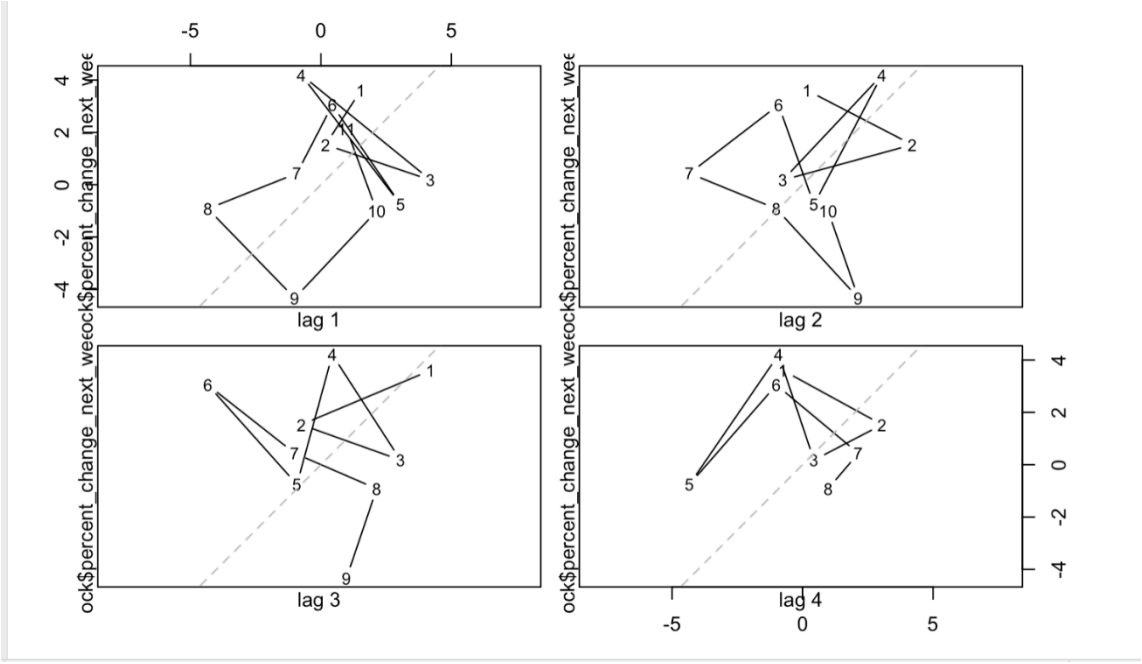
| Model <chr> | MSE <dbl> |
|---|---|
| Linear Regression | 9.130437 |
| Decision Tree | 9.147271 |
| SVR | 8.439303 |

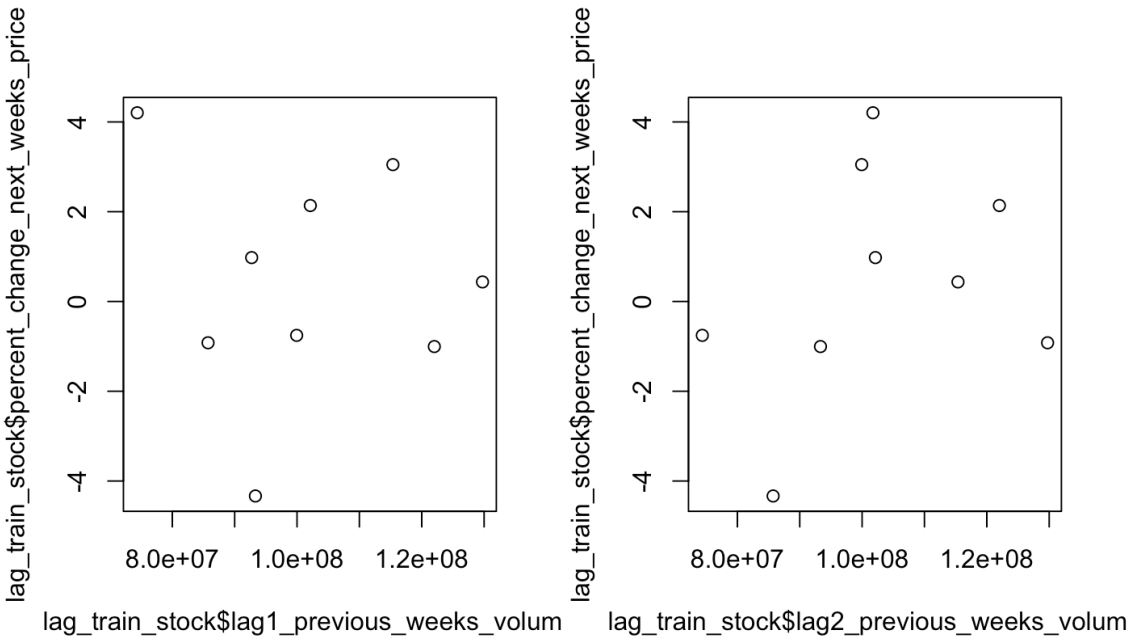
Based on the MSEs, the SVR outperformed the other models, offering the most accurate predictions for stock price changes. Linear Regression provided valuable insights into the most significant factors affecting price changes, while Decision Tree Regression was useful for capturing non-linear interactions, though it didn't perform as well in terms of prediction accuracy.

Stepwise regression was applied to the linear model to identify the most significant predictors of stock price changes, but the result was a drastic increase in average MSE, indicating that our initial predictor of *previous_weeks_volume* was better.

| Model<br><chr> | MSE<br><dbl> |
|---|---|
| Linear Regression | 9.130437 |
| Decision Tree | 9.147271 |
| SVR | 8.439303 |
| Step LM | 3785.611966 |

Lag variables were also introduced to account for temporal dependencies in stock price changes. We explored the effects of lag periods 1 week through 4 for all 30 stocks, but there was no significant influence of any of the lags on *percent_change_next_weeks_price*.






To assess the risk of individual stocks, we computed beta coefficients using the Capital Asset Pricing Model (CAPM). The beta values provided insights into relative volatility compared to the overall market. Stocks with higher beta values (above 1) tend to exhibit more volatility, while those with lower beta values (below 1) are more stable. This indicated that the weekly returns for stable stocks exhibit low volatility, and their price fluctuations would only be a fraction from what historical data showed. In other words,

their weekly returns would most likely behave the same way in the future if their beta value was below 1. If a stock lost money the past week, it will lose money the following week. Below are the beta values for each stock along with their average return and standard deviation.

| Stock <chr> | Beta <dbl> | Stock <chr> | Beta <dbl> | Stock <chr> | Beta <dbl> |
|---|---|---|---|---|---|
| AA | 1.2996180 | HD | 0.9854805 | MRK | 0.5415464 |
| AXP | 1.0879594 | HPQ | 1.4221849 | MSFT | 0.7130323 |
| BA | 1.6229882 | IBM | 0.9208968 | PFE | 0.7249845 |
| BAC | 0.6510668 | INTC | 1.2509146 | PG | 0.4849129 |
| CAT | 1.4917179 | JNJ | 0.7594810 | T | 0.8039250 |
| CSCO | 0.6541192 | JPM | 0.8660191 | TRV | 0.9988462 |
| CVX | 0.8431656 | KRFT | 0.1895133 | UTX | 1.1975557 |
| DD | 1.1270510 | KO | 0.6797966 | VZ | 0.7601467 |
| DIS | 1.4847032 | MCD | 0.7435486 | WMT | 0.6854153 |
| GE | 1.3526311 | MMM | 1.2372817 | XOM | 1.1963249 |

| | returns_mean | returns_sd |
|---|---|---|
| AA | -0.0025562614 | 0.03450483 |
| AXP | 0.0039787288 | 0.02867944 |
| BA | 0.0014706601 | 0.02740461 |
| BAC | -0.0120307527 | 0.03342655 |
| CAT | 0.0032371593 | 0.03336199 |
| CSCO | -0.0132726501 | 0.03920114 |
| CVX | 0.0032523639 | 0.02448295 |
| DD | 0.0021434643 | 0.02664204 |
| DIS | -0.0016628865 | 0.02746833 |
| GE | -0.0007380844 | 0.02560781 |
| HD | 0.0011017053 | 0.02335630 |
| HPQ | -0.0098428976 | 0.03914964 |
| IBM | 0.0047343827 | 0.01806670 |
| INTC | 0.0015586246 | 0.03211640 |
| JNJ | 0.0018110711 | 0.02068670 |
| JPM | -0.0039127044 | 0.02246357 |
| KRFT | 0.0044823469 | 0.01777007 |
| KO | 0.0014376506 | 0.01628497 |
| MCD | 0.0041737383 | 0.01919738 |
| MMM | 0.0023999818 | 0.02069039 |
| MRK | -0.0028985966 | 0.02654569 |
| MSFT | -0.0065998397 | 0.01858821 |
| PFE | 0.0041023671 | 0.02598915 |
| PG | -0.0011006261 | 0.01777888 |
| T | 0.0024128625 | 0.01926273 |
| TRV | 0.0027203616 | 0.01934159 |
| UTX | 0.0028884330 | 0.02084871 |
| VZ | 0.0002326181 | 0.01783121 |
| WMT | -0.0011305653 | 0.01894121 |
| XOM | 0.0009597025 | 0.02533114 |
| DJI | 0.0010167236 | 0.01440911 |

Lastly, SVR was used to predict stock values measured against Q2 values and the MSEs for each stock were found to assess prediction error. KO stock predictions had the least amount of error while CAT had the highest error rate.

| Stock <chr> | MSE <dbl> | Stock <chr> | MSE <dbl> | Stock <chr> | MSE <dbl> |
|---|---|---|---|---|---|
| KO | 2.011054 | BAC | 5.958973 | MSFT | 10.454202 |
| T | 2.022104 | MMM | 6.053698 | AXP | 10.877980 |
| WMT | 2.295598 | JNJ | 6.319670 | CVX | 11.257649 |
| MCD | 2.982867 | GE | 6.928553 | XOM | 11.318532 |
| KRFT | 2.986695 | PFE | 7.189441 | CSCO | 12.326586 |
| IBM | 3.031101 | TRV | 7.291856 | DIS | 13.086128 |
| VZ | 5.026785 | JPM | 7.571263 | INTC | 14.199438 |
| MRK | 5.234445 | BA | 7.946320 | AA | 16.864395 |
| PG | 5.476923 | UTX | 9.406524 | HPQ | 17.522851 |
| HD | 5.636685 | DD | 10.442583 | CAT | 23.458176 |

## Monte Carlo Simulation and Its Added Value

In addition to the deterministic models used in this study—Linear Regression, Support Vector Regression, and Decision Trees—we incorporated a Monte Carlo simulation model to provide a probabilistic perspective on future stock behavior. Unlike the machine learning models which output a single predicted value, the Monte Carlo approach simulates thousands of possible future price paths based on historical return distributions, enabling us to estimate a range of outcomes, including best- and worst-case scenarios.

The Monte Carlo model offers a complementary dimension to the earlier models by capturing the inherent uncertainty and randomness in financial markets. While SVR excelled in accuracy (as measured by MSE), it provides a point estimate without quantifying the probability or likelihood of outcomes. Monte Carlo simulation, on the other hand, produces confidence intervals and downside risk estimates such as Value at Risk (VaR), enhancing our understanding of potential variability and tail risks for each stock.

A tibble: 15 × 4

| stock <fctr> | mean_return <dbl> | sd <dbl> | var_95 <dbl> |
|---|---|---|---|
| CSCO | -0.119671926 | 0.13028200 | -0.3083320 |
| BAC | -0.103614356 | 0.09934831 | -0.2578544 |
| HPQ | -0.060320495 | 0.12502445 | -0.2553304 |
| AA | -0.025223323 | 0.13068723 | -0.2378083 |
| MSFT | -0.031738354 | 0.08306532 | -0.1591469 |
| MRK | -0.004065667 | 0.09165127 | -0.1462863 |
| GE | 0.005949118 | 0.08790020 | -0.1368768 |
| CAT | 0.062845843 | 0.13434045 | -0.1365956 |
| DIS | 0.025800482 | 0.10404301 | -0.1330774 |
| JPM | 0.013370020 | 0.09204759 | -0.1314208 |

1-10 of 15 rows

Previous 1 2 Next

## Comparison and Integration with Other Models

Linear Regression offered interpretability but lacked flexibility in capturing non-linear patterns. Decision Trees addressed non-linearity but struggled with overfitting. SVR balanced these trade-offs and proved most accurate for predicting stock price changes. However, all three models assume a fixed structure and are sensitive to historical training data. Monte Carlo simulation differs in that it builds on historical return distributions without rigid model structures, simulating stochastic outcomes under uncertainty. This makes it particularly useful for scenario analysis and risk management.

Moreover, the Monte Carlo approach supported and validated the CAPM-based risk analysis by empirically illustrating how downside risk behaves across different stocks. For example, stocks with lower beta values under CAPM consistently showed narrower Monte Carlo simulation intervals and lower VaR, providing empirical backing to the theoretical risk assessment. This cross-validation strengthens our confidence in both the beta estimates and the overall risk framework used.

## Why Monte Carlo Strengthens the Analysis

The inclusion of Monte Carlo modeling adds depth and robustness to the case study. It not only verifies patterns identified by SVR and CAPM but also provides actionable insights into the likelihood of losses. In practical financial decision-making, knowing that a stock has a high expected return is useful—but knowing the probability that it could lose 10% over the next quarter is even more valuable. By quantifying uncertainty and tail risk, the Monte Carlo simulation gives stakeholders a more comprehensive view of reward-to-risk trade-offs, thereby enhancing the decision-making process.

## The Monte Carlo Outcome

The output from the Monte Carlo provides three scores, and the ideal stock would have a high Mean, with a low SD and Variance.  Looking at the list of stocks, we find the top five stocks to purchase are (in order): KRFT, IBM, MDC, KO, and TUX and the worst five stocks to purchase are MRK, BAC, HPQ, AA, AND CSCO. Although this leaves out important information like price, dividend, and some other fundamentals like P/E, P/S, and P/B, this still gives us usable information to compare to the other models.  Then comparing the outputs we can make a better informed decision on which stocks to purchase based on value and stability, or which to avoid.

## Conclusion

This study demonstrates that machine learning techniques can predict stock price changes in the Dow Jones Index. The SVR was the best-performing model, while Linear Regression provided valuable insights into the predictors of stock price changes. CAPM was useful for assessing stock risk. For future improvements, incorporating deep learning methods and macroeconomic indicators could enhance prediction accuracy. Additionally, exploring alternative time-series forecasting models may provide further insights into stock price dynamics.

## Works Cited


Abraham, Bovas, and Johannes Ledolter. *Statistical methods for forecasting*. John Wiley & Sons, 2009.

Bansal, Malti, et al. “Stock Market Prediction with High Accuracy Using Machine Learning Techniques.” *Procedia Computer Science*, vol. 215, Elsevier, Amsterdam, 2022, pp. 247–265.

Breiman, Leo, et al. *Classification and regression trees*. Routledge, 2017.

Smola, Alex J, and Bernhard Scholkopf. Kluwer Academic Publishers, 2004, pp. 1–24, *A Tutorial on Support Vector Regression*.


## Appendix

```{r}
library(tidyverse)
library(readr)
library(e1071)
library(tree)
library(caret)
```

```{r}
dow_data = read_csv("dow_jones_index.data")
```

```
Rows: 750 Columns: 16
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (8): stock, date, open, high, low, close, next_weeks_open, next_weeks_close
dbl (8): quarter, volume, percent_change_price, percent_change_volume_over_last_wk, pr ...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```{r}
str(dow_data)
```

```{r}
# Remove '$' and ',' from price columns and convert to numeric
price_cols ← c("open", "high", "low", "close", "next_weeks_open", "next_weeks_close")

dow_data ← dow_data %>%
  mutate(across(all_of(price_cols), ~ as.numeric(gsub("[$,]", "", .))))

# Fix percentage columns
dow_data ← dow_data %>%
  mutate(across(starts_with("percent_change"), as.numeric))

# Convert categorical variables
dow_data$quarter ← as.factor(dow_data$quarter)
dow_data$stock ← as.factor(dow_data$stock)
dow_data$date = as.Date(dow_data$date, format = "%m/%d/%Y")

```

```{r}
str(dow_data)
```

```
tibble [750 × 16] (S3: tbl_df/tbl/data.frame)
 $ quarter                          : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 1 ...
 $ stock                            : Factor w/ 30 levels "AA","AXP","BA",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ date                             : Date[1:750], format: "2011-01-07" "2011-01-14" ...
 $ open                             : num [1:750] 15.8 16.7 16.2 15.9 16.2 ...
 $ high                             : num [1:750] 16.7 16.7 16.4 16.6 17.4 ...
 $ low                              : num [1:750] 15.8 15.6 15.6 15.8 16.2 ...
 $ close                            : num [1:750] 16.4 16 15.8 16.1 17.1 ...
 $ volume                           : num [1:750] 2.40e+08 2.43e+08 1.38e+08 1.51e+08 1.54e+08 ...
 $ percent_change_price             : num [1:750] 3.79 -4.43 -2.47 1.64 5.93 ...
 $ percent_change_volume_over_last_wk: num [1:750] NA 1.38 -43.02 9.36 1.99 ...
```

```
 $ percent_change_volume_over_last_wk: num [1:750] NA 1.38 -43.02 9.36 1.99 ...
 $ previous_weeks_volume             : num [1:750] NA 2.40e+08 2.43e+08 1.38e+08 1.51e+08 ...
 $ next_weeks_open                   : num [1:750] 16.7 16.2 15.9 16.2 17.3 ...
 $ next_weeks_close                  : num [1:750] 16 15.8 16.1 17.1 17.4 ...
 $ percent_change_next_weeks_price   : num [1:750] -4.428 -2.471 1.638 5.933 0.231 ...
 $ days_to_next_dividend             : num [1:750] 26 19 12 5 97 90 83 76 69 62 ...
 $ percent_return_next_dividend      : num [1:750] 0.183 0.188 0.19 0.186 0.175 ...
```

```{r}
stocks = unique(dow_data$stock)
```

## Create a linear model for each stock

```{r}
columns= c("Stock","MSE")
lm_acc_df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(lm_acc_df) = columns


for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
  train_stock = stock_df %>%
    filter(quarter == 1)
  test_stock = stock_df %>%
    filter(quarter == 2)
  lm_fit = lm(percent_change_next_weeks_price ~ previous_weeks_volume,
              data = train_stock)
  lm_preds  <- predict(lm_fit, test_stock)
  lm_mse  <- mean((lm_preds - test_stock$percent_change_next_weeks_price)^2)
  vec = c(i, lm_mse)
  lm_acc_df[i, ] = vec
```

## Create a svr model for each stock

```{r}
columns= c("Stock","MSE")
svr_acc_df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(svr_acc_df) = columns


for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
  train_stock = stock_df %>%
    filter(quarter == 1)
  test_stock = stock_df %>%
    filter(quarter == 2)
  svr_fit = svm(percent_change_next_weeks_price ~ previous_weeks_volume,
                data = train_stock, kernel = "radial")
  svr_preds  <- predict(svr_fit, test_stock)
  svr_mse  <- mean((svr_preds - test_stock$percent_change_next_weeks_price)^2)
  vec = c(i, svr_mse)
  svr_acc_df[i, ] = vec
}
```

## Create a decision tree model for each stock

```{r}
columns= c("Stock","MSE")
tree_acc_df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(tree_acc_df) = columns


for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
```

## Create a decision tree model for each stock

```{r}
columns= c("Stock","MSE")
tree_acc_df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(tree_acc_df) = columns


for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
  train_stock = stock_df %>%
    filter(quarter == 1)
  test_stock = stock_df %>%
    filter(quarter == 2)
  tree_fit = tree(percent_change_next_weeks_price ~ previous_weeks_volume,
                  data = train_stock, method = "anova")
  tree_preds  <- predict(tree_fit, test_stock)
  tree_mse  <- mean((tree_preds - test_stock$percent_change_next_weeks_price)^2)
  vec = c(i, tree_mse)
  tree_acc_df[i, ] = vec
}
```

## See which model is best

```{r}
lm_mse = mean(as.numeric(lm_acc_df$MSE))
svr_mse = mean(as.numeric(svr_acc_df$MSE))
tree_mse = mean(as.numeric(tree_acc_df$MSE))

results <- tibble(
  Model = c("Linear Regression", "Decision Tree", "SVR"),
  MSE = c(lm_mse, tree_mse, svr_mse)
```


## See which model is best

```{r}
lm_mse = mean(as.numeric(lm_acc_df$MSE))
svr_mse = mean(as.numeric(svr_acc_df$MSE))
tree_mse = mean(as.numeric(tree_acc_df$MSE))

results <- tibble(
  Model = c("Linear Regression", "Decision Tree", "SVR"),
  MSE = c(lm_mse, tree_mse, svr_mse)
)

print(results)
```


A tibble: 3 × 2

| Model <chr> | MSE <dbl> |
|---|---|
| Linear Regression | 9.130437 |
| Decision Tree | 9.147271 |
| SVR | 8.439303 |

3 rows

## Create a linear stepwise model for each stock

```{r results='hide'}
columns= c("Stock","MSE")
step_lm_acc_df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(step_lm_acc_df) = columns
```

## Create a linear stepwise model for each stock

```{r results='hide'}
columns= c("Stock","MSE")
step_lm_acc_df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(step_lm_acc_df) = columns


for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
  stock_df = na.omit(stock_df)
  train_stock = stock_df %>%
    filter(quarter == 1)
  test_stock = stock_df %>%
    filter(quarter == 2)
  lm_fit = lm(percent_change_next_weeks_price ~ percent_change_price +
                percent_change_volume_over_last_wk + days_to_next_dividend +
                percent_return_next_dividend + high + low + close + volume +
                previous_weeks_volume,
                data = train_stock)
  step_lm = step(lm_fit, direction = "both")
  lm_preds  <- predict(step_lm, test_stock)
  lm_mse  <- mean((lm_preds - test_stock$percent_change_next_weeks_price)^2)
  vec = c(i, lm_mse)
  step_lm_acc_df[i, ] = vec
}
```

## See which model is best

```{r}
lm_mse = mean(as.numeric(lm_acc_df$MSE))
step_lm_mse = mean(as.numeric(step_lm_acc_df$MSE))
svr_mse = mean(as.numeric(svr_acc_df$MSE))
tree_mse = mean(as.numeric(tree_acc_df$MSE))

results <- tibble(
  Model = c("Linear Regression", "Decision Tree", "SVR", "Step LM"),
  MSE = c(lm_mse, tree_mse, svr_mse, step_lm_mse)
)

print(results)
```

A tibble: 4 × 2

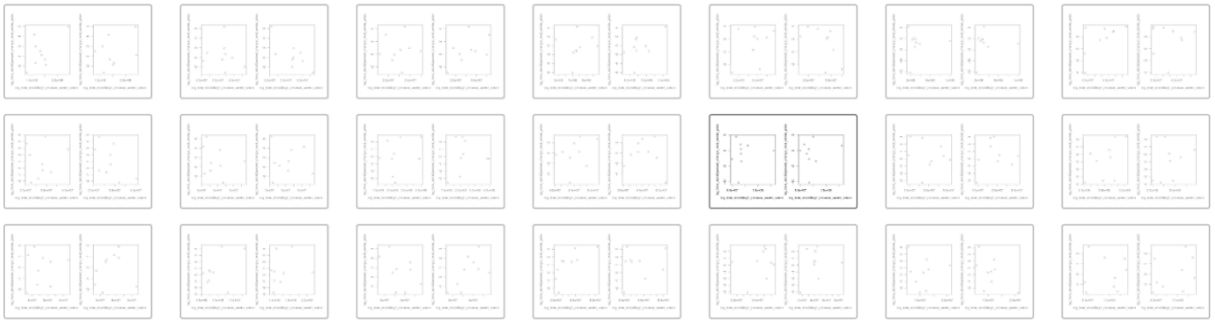
| Model <chr> | MSE <dbl> |
|---|---|
| Linear Regression | 9.130437 |
| Decision Tree | 9.147271 |
| SVR | 8.439303 |
| Step LM | 3785.611966 |

4 rows

## Create Lag variables

```{r}
for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
  train_stock = stock_df %>%
    filter(quarter == 1)
  test_stock = stock_df %>%
    filter(quarter == 2)
  lag.plot(train_stock$percent_change_next_weeks_price, set.lags = 1:4)
}
```

```{r}
for (i in stocks) {
  stock_df = dow_data %>%
    filter(dow_data$stock == i)
  train_stock = stock_df %>%
    filter(quarter == 1)
  test_stock = stock_df %>%
    filter(quarter == 2)
  lag_train_stock = train_stock %>%
    mutate(
      lag1_previous_weeks_volume = lag(previous_weeks_volume, 1),
      lag2_previous_weeks_volume = lag(previous_weeks_volume, 2)
      )
  lag_train_stock = na.omit(lag_train_stock)
  par(mfrow = c(1,2))
  plot(y = lag_train_stock$percent_change_next_weeks_price, x = lag_train_stock$lag1_previous_weeks_volume)
  plot(y = lag_train_stock$percent_change_next_weeks_price, x = lag_train_stock$lag2_previous_weeks_volume)
}
```



## CAPM

```{r}
dji = read_csv("DJI_CaseStudy3.csv")
```

```
New names:
• `` -> `...7`
Rows: 33 Columns: 7
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
num  (5): Open, High, Low, Close , Adj Close
lgl  (1): ...7
date (1): Date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```{r}
dji = dji[,c(1:6)]
dji = na.omit(dji)
```

```{r}
dji_df = as.data.frame(dji[,])
```

```{r}
return_dji = na.omit(quantmod::Delt(dji_df[,5]))
```

```{r}
return_df = data.frame(matrix(nrow = 24, ncol = 0))

for (i in stocks) {
  stock_df = dow_data %>%
  filter(dow_data$stock == i)
  stock_df = stock_df %>%
    select(open, high, low, close)
  stock_df = as.data.frame(stock_df)
  return = na.omit(quantmod::Delt(stock_df[,4]))
  return_df = cbind(return_df, return)
}
```

```{r}
return_data = cbind(return_df, return_dji)
colnames(return_data) = c(lm_acc_df$Stock, "DJI")
```

```{r}
returns_mean = apply(return_data,2,mean)
returns_sd = apply(return_data,2,sd)
cbind(returns_mean,returns_sd)
```

#### Calculate Betas

```{r}
calculate_betas <- function(return_data) {
  columns <- c("Stock", "Beta")
  betas <- data.frame(matrix(nrow = 0, ncol = length(columns)))
  colnames(betas) <- columns

  stock_names <- colnames(return_data)[colnames(return_data) != "DJI"]

  for (stock in stock_names) {
    regression_data <- data.frame(
      stock_return = return_data[[stock]],
      market_return = return_data$DJI
    )

    #lm_model <- lm(stock_return ~ market_return, data = regression_data)
    svr_model <- svm(stock_return ~ market_return, data = regression_data, kernel = "linear")
    beta <- coef(svr_model)[2]
    betas <- rbind(betas, data.frame(Stock = stock, Beta = beta))
  }
  rownames(betas) = NULL

  return(betas)
}

betas <- calculate_betas(return_data)
print(betas)
```

#### SVR for predictions

```{r}
columns <- c("Stock", "MSE")
svr_acc_df <- data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(svr_acc_df) <- columns

for (stock_name in stocks) {
  stock_df <- dow_data %>%
    filter(stock == stock_name)

  train_stock <- stock_df %>%
    filter(quarter == 1)
  test_stock <- stock_df %>%
    filter(quarter == 2)

  svr_fit <- svm(percent_change_next_weeks_price ~ previous_weeks_volume,
                 data = train_stock, kernel = "radial")

  svr_preds <- predict(svr_fit, test_stock)

  svr_mse <- mean((svr_preds - test_stock$percent_change_next_weeks_price)^2)

  svr_acc_df <- rbind(svr_acc_df, data.frame(Stock = stock_name, MSE = svr_mse))
}

# Convert MSE column to numeric (if needed)
svr_acc_df$MSE <- as.numeric(as.character(svr_acc_df$MSE))

# Display results, sorted by MSE
svr_acc_df <- svr_acc_df %>% arrange(MSE)
```