# Predicting Stock Returns of the S&P 500

MIST 7770 Final Project

*By: Tucker Tracy, Alessio Raggi, Katie McCallum, Stefan Tse, and Brian Zeldin*

Terry College of Business
UNIVERSITY OF GEORGIA

# Context & Questions

- Investors trade an average of $18.9 billion per day on the NYSE
- Investors want to know when to invest and when to sell stocks so they can make money
- The dataset consists of a large collection of stock data from many companies across multiple industries, all in the S&P 500
- By incorporating a wide array of variables across industries, the dataset enables the creation of models to analyze future stock behavior

- QUESTION: What is the average stock return for companies in the S&P 500 for future years?

# Data & Variables

- The dataset is from Kaggle.
- We believe that this data is adequate to address our problem because it gives both financial statement variables as well as stock return data. These seem to be the most important datapoints when calculating returns on a stock
- 81 variables metrics extracted from annual SEC 10K filings (2012-2016)
  - Key variables include total revenue, net income, and EBIT, offering insights into a company's profitability and future stock prices
- 851,000 observations of daily price data for the S&P 500 over five years
  - Daily highs, lows, open, and close are given

# Data & Variables

- Merged the financial data and the daily pricing data
- After merging, there were 81 variables and 1,220 observations (down from over 800,000 observations)
  - Each observation was a S&P 500 stock during a certain year
- No leakage!
  - Used the previous year's data when performing the join to avoid leakage
  - If we had joined our price data from 2012 w company data from 2011, that would've been leakage. However, we merged on the previous years data to prevent this

# Data & Variables

- **Target variable:** Average return of each company in the S&P 500 for each year
  - This is a numeric type of variable
  - Conservative return = [(close-open)/ close] on average for each year
  - Optimal return = [(high-low)/ high] on average for each year
  - Ultimately decided to use the conservative return because we thought it would be more accurate and more like real- world returns
- **Predictors/ important features:**
  - Date variables are used for all the stock observations. The open, close, high, and low are all for the day. These were averaged across each year
  - All features were used in the random forest to create more diverse decision tree branches
  - Most important features of linear model: year, cash ratio, net cash flow, operating margin, minority interest, changes in inventories, accounts receivables, average open price

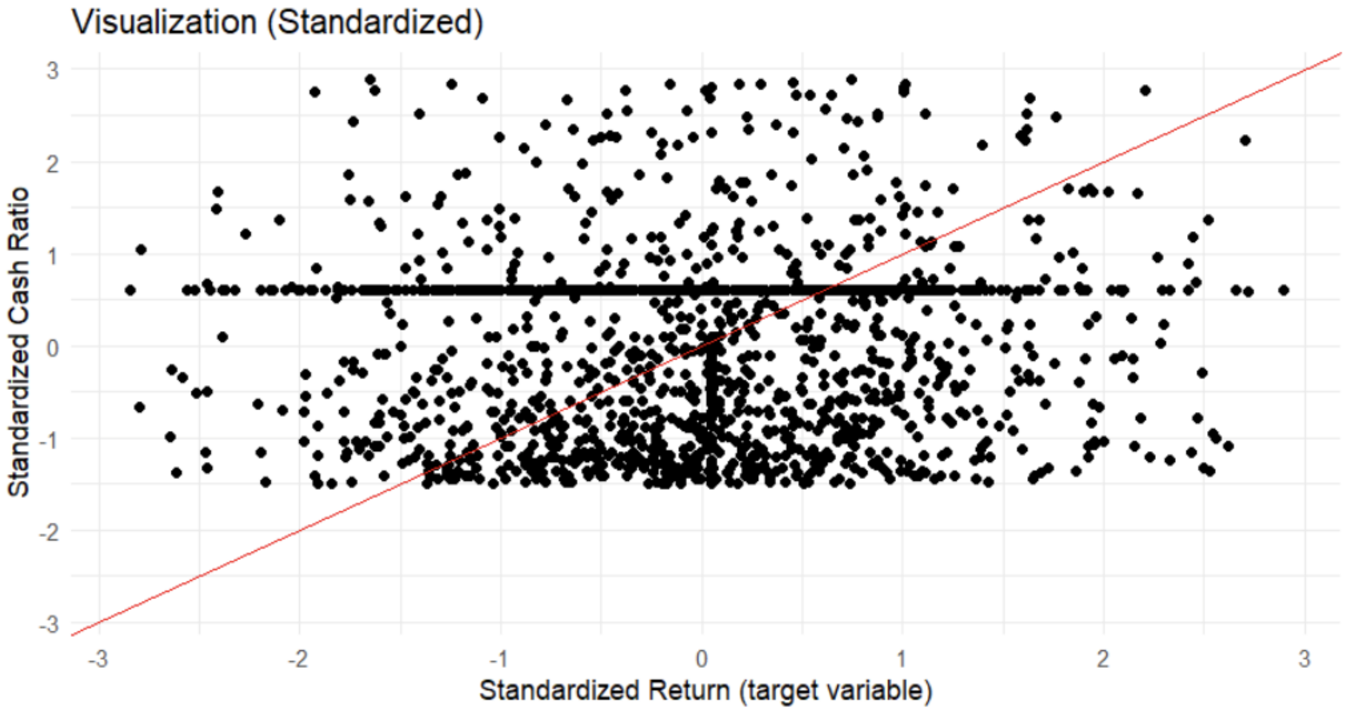
# Data & Variables

| | Year | Cash Ratio | Net Cash Flow | Operating Margin | Minority Interest | Changes in Inventories | Accounts Receivables | Average Open Price |
|---|---|---|---|---|---|---|---|---|
| Mean | 2,013.712 | 51.940 | 33,785,262 | 16.063 | 97,122,016 | -24,329,391 | -36,295,399 | 61.892 |
| Median | 2,014 | 55 | 49,787,717 | 15 | 1,800,000 | 0 | -36,500,000 | 59.158 |
| Standard Deviation | 1.039 | 34.594 | 247,808,590 | 9.522 | 166,860,351 | 39,142,812 | 69,839,194 | 30.647 |

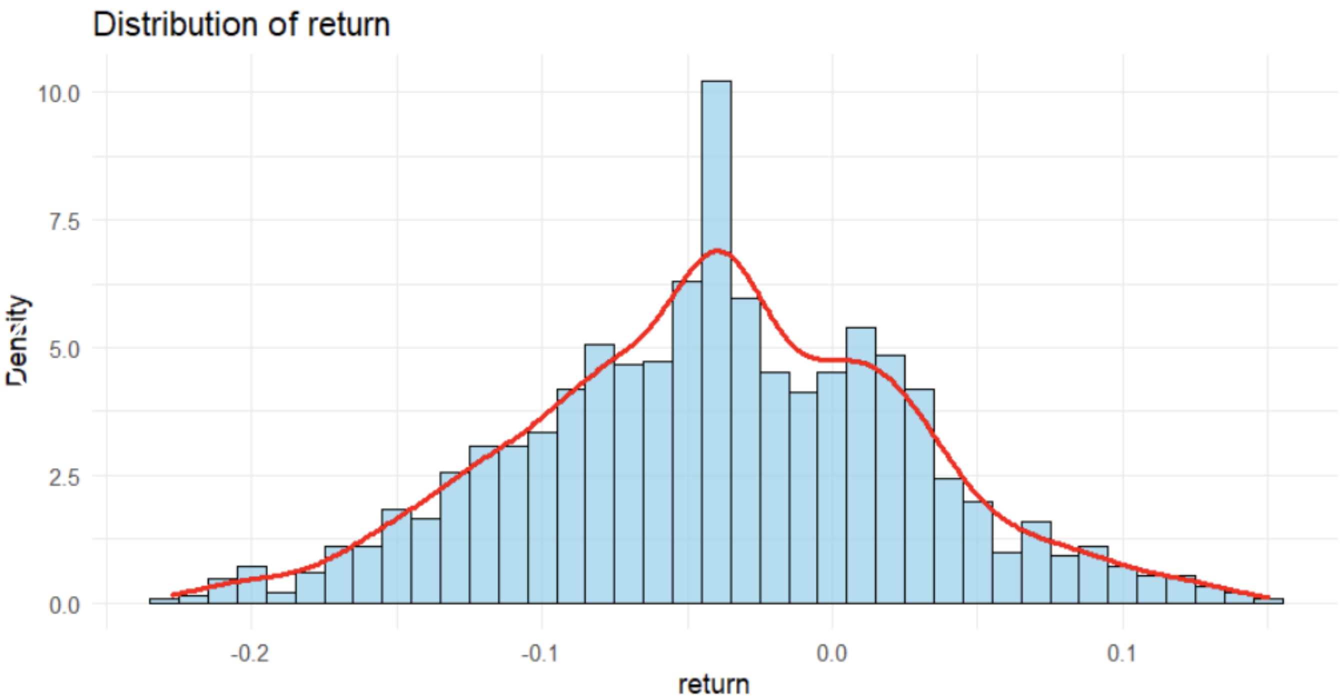
# Data & Variables

- Cannot simply remove missing data and outliers because it will remove too much of the data
- Our data contained both missing values and outliers
- For the missing data, the simple average for each variable was calculated and used to impute the missing values
- For the outliers, the interquartile range was calculated. An if statement was created so that if a datapoint was outside the range, it was replaced with either the lower or upper bound, depending on which was closer. If the datapoint was within the range, the data remained unchanged

# Descriptive Visualizations



# Descriptive Visualizations



# Method – Data Prep

- Started by getting the average return for each stock for each year
- Then merged the two sets of data, fund (company annual financial statements) and prices (everyday stock market information)
    - Joined the two files on the previous year, so that there was no leakage
    - Replaced missing data with the average and the outliers using the IQR method
- Next, the data was scaled to address issues with coefficient estimates being skewed by differences in data magnitude
    - Standardized with z- scores ($z = (X_i - \mu) / \sigma$)

# Method – Data Prep

- Once the previous steps were completed, a glaring issue with the data was realized. There were 1,220 observations, but only 81 variables
  - Rule of thumb: for every 10 observations, you must have 1 predictor
- To fix this issue, LASSO regression was used.
  - Pushed coefficients that are not predictors to zero
  - This acts as a method for feature selection
  - Selected the variables with the highest coefficients in absolute magnitude

# Method – Data Prep

STEP 1:

```
# Compute conservative and optimal return
prices <- prices %>% mutate(conservative_return =
((open-close)/open)*100,
                            optimal_return = ((high-
low)/low)*100,
                            month = month(date),
                            year = year(date),
                            year_prev = year - 1) %>%
  filter(!is.na(date))

# Group the prices data by symbol and year
prices_grouped <- prices %>%
group_by(symbol,year) %>%
  summarise(open = mean(open),
          close = mean(close),
          low = mean(low),
          high = mean(high),
          volume = mean(volume))

# Join company data with basic mean price data on
the same year
first_join <- merge(fundamentals,prices_grouped,
                    by.x = c("Ticker Symbol","year"),
                    by.y = c("symbol","year"))

# Group the prices data by symbol and year
# and calculate the mean conservative and optimal
return
return_grouped <- prices %>%
group_by(symbol,year) %>%
  summarise(mean_conservative_return =
mean(conservative_return),
            mean_optimal_return =
mean(optimal_return)) %>%
  mutate(year_prev = year - 1)

# Join current year return data with the previous
year company data
second_join <- merge(first_join,return_grouped,
                    by.x = c("Ticker Symbol","year"),
                    by.y = c("symbol","year_prev"))

data <- second_join %>% select(c(1,2,4:85,87,88))
```

# Method – Data Prep

STEP 2:

```
basic_variables <- data %>% select(c(2,4:76,78:86))

# Clean the column names by removing spaces and special characters
clean_colnames <- function(x) {
  gsub("[^A-Za-z0-9]", "", x)
}

# Apply the function to clean the column names
colnames(basic_variables) <- clean_colnames(colnames(basic_variables))

# Display cleaned column names
colnames(basic_variables)

# MISSING VALUE TIME
impute_missing <- function(data) {
  for(i in seq_len(ncol(data))) {
    column_mean <- mean(data[,i], na.rm = TRUE)
    data[,i] <- ifelse(is.na(data[,i]), column_mean, data[,i])
  }
  return(data)
}

no_missing <- impute_missing(basic_variables)
sum(is.na(no_missing))

# OUTLIER TIME

## Replace mean with only mean of respective companies ##
outlier_detect <- function(data) {
  for(i in seq_len(ncol(data))) {
    q1 <- quantile(data[,i], 0.25)
    q3 <- quantile(data[,i], 0.75)
    iqr <- q3 - q1

    upper_threshold <- q3 + (1.5 * iqr)
    lower_threshold <- q1 - (1.5 * iqr)

    # Calculate the column mean before handling outliers
    column_mean <- mean(data[,i], na.rm = TRUE)

    # Replace outliers with the mean
    data[,i] <- ifelse(data[,i] > upper_threshold | data[,i] < lower_threshold,
                       column_mean, data[,i])
  }
  return(data)
}

no_outliers_or_missing <- outlier_detect(no_missing)
```

# Method – Data Prep

STEP 3:

```
# scaling
predictors <- no_outliers_or_missing %>% select(c(1:81))
write_csv(predictors,file = "predictors.csv")
conservative_target <- no_outliers_or_missing %>% select(82)
scaled_predictors <- as.data.frame(scale(predictors))

# training
train_indices <- c(1:(nrow(scaled_predictors)*0.8))
conservative_train_X <- scaled_predictors[train_indices,]
conservative_train_y <- conservative_target[train_indices,]
conservative_train <- cbind(conservative_train_X,conservative_train_y)
conservative_train <- as.data.frame(conservative_train)

# testing
conservative_test_X <- scaled_predictors[-train_indices,]
conservative_test_y <- conservative_target[-train_indices,]
conservative_test <- cbind(conservative_test_X,conservative_test_y)
conservative_test <- as.data.frame(conservative_test)

# write out the data for conservative return
write_csv(conservative_train,file = "conservative_train.csv")
write_csv(conservative_test,file = "conservative_test.csv")
```

# Method – Machine Learning

- Two prediction models were used:
  - Random Forest
    - There is no assumption about the nature of the data
    - This is appropriate for this data set because random forests are great at handling complexity. These models reduce overfitting while capturing important features, making it great for volatile stock data
  - Linear Regression
    - Assumes there is a linear relationship between the variables
    - This is a good model for this data because it is easy to interpret the data, such as historical prices. It also provides a clear understanding of the magnitude and direction of the variables
- Split the data for training and testing
  - 80% training & 20% testing
- Evaluated performance using Mean Absolute Error (MAE) and Mean Absolute Percentage Error

# Method – Machine Learning

RANDOM FOREST:

```
# Train random forest including all predictors
rf <- randomForest(conservative_train_y~., data=conservative_train, proximity=TRUE)
print(rf)

# Make predictions with random forest on test data
y_predicted <- predict(rf, conservative_test_X)

# Create data frame with random forest predictions and actual test values
pred_v_actual <- as.data.frame(cbind(y_predicted,conservative_test_y))

# Create evaluation data frame
eval <- pred_v_actual %>%
  mutate(diff = conservative_test_y - y_predicted,
         squared_error = (conservative_test_y - y_predicted)^2,
         abs_error = abs(conservative_test_y - y_predicted),
         mape = abs(diff)/conservative_test_y)

# Print evaluation metrics
paste("Random Forest Mean Absolute Error:" {mean(eval$abs_error)})
paste("Random Forest Mean Absolute Percentage Error:" {mean(eval$mape)})
```

# Method – Machine Learning

LASSO MODEL:

```
# Perform cross-validation to find best lambda penalty value
cv_model <- cv.glmnet(x = as.matrix(conservative_train_X), y = as.matrix(conservative_train_y),
                  alpha = 1, maxit = 10000000)

# Save lambda that minimizes MSE
best_lambda <- cv_model$lambda.min

# Plot MSE of model as function of lambda values
plot(cv_model)

# Save best model
lasso_model <- glmnet(as.matrix(conservative_train_X), as.matrix(conservative_train_y),
                  alpha = 1, lambda = best_lambda)

# Print beta coefficients
beta_coefficients <- coef(lasso_model)
print(beta_coefficients[order(abs(beta_coefficients), decreasing=TRUE),])
```

```
# Make predictions on test data
y_predicted <- predict(lasso_model, as.matrix(conservative_test_X))

pred_v_actual <- as.data.frame(cbind(y_predicted,conservative_test_y))

# Create evaluation data frame
eval <- pred_v_actual %>%
  mutate(diff = conservative_test_y - y_predicted,
         squared_error = (conservative_test_y - y_predicted)^2,
         abs_error = abs(conservative_test_y - y_predicted),
         mape = abs(diff)/conservative_test_y)

paste("Lasso Regression Mean Absolute Error:" {mean(eval$abs_error)})
paste("Lasso Regression Mean Absolute Percentage Error:" {mean(eval$mape)})
```

# Method – Machine Learning

LINEAR MODEL:

```
# Train linear model on select predictors
linear_model <- lm(conservative_train_y ~ year + CashRatio + NetCashFlow +
                   OperatingMargin + MinorityInterest + ChangesinInventories +
                   AccountsReceivable + open, data = conservative_train)

summary(linear_model)

# Make predictions on test data
y_predicted <- predict(linear_model, conservative_test_X)

pred_v_actual <- as.data.frame(cbind(y_predicted,conservative_test_y))

# Create evaluation data frame
eval <- pred_v_actual %>%
  mutate(diff = conservative_test_y - y_predicted,
         squared_error = (conservative_test_y - y_predicted)^2,
         abs_error = abs(conservative_test_y - y_predicted),
         mape = abs(diff)/conservative_test_y)

# Print out evaluation metrics
paste("Linear Mean Absolute Error:" {mean(eval$abs_error)})
paste("Linear Mean Absolute Percentage Error:" {mean(eval$mape)})
```

# Evaluation/ Results

- Linear
  - **MSE: 2.389**
  - MAE: 0.052
  - MAPE: 0.183%
  - The mean absolute percentage error (MAPE) indicates that this model's predictions differ by 0.183% from the actual values. This suggests that this model produces relatively accurate predictions in terms of percentage error
  - The MSE is a bit higher though, suggesting that this model may not be perfect for all patterns in the stock data, especially the non-linear ones
- Random forest
  - **MSE: 0.138**
  - MAE: 0.047
  - MAPE: 0.322%
  - % Variance explained: 44.39%
  - This model shows higher predictive accuracy
  - With an MSE of 0.138 and an MAE 0.047, the model shows smaller deviations between the predicted and actual values. The MAPE is higher, which indicates that its relative error (as opposed to absolute errors) is higher. This *could* mean that the model could deviate more on a percentage basis for some models
  - The percentage of variance explained is 44.39%, which means that 44.39% of the variability in stock returns is accounted for by the model. This indicates that the random forest captures some patterns in the data, but cannot capture them all. This is no surprise because predicting this output has many external factors that cannot be accounted for always (elections, market trends, etc.)

- If MSE or MAE is the deciding factor, choose Model 2, as it provides more accurate absolute predictions
- If MAPE is more important, choose Model 1, as it performs better with percentage errors

# Limitations

It is very hard to predict the stock market. The following things were not included in the model:

- Competition
- Politics
- Market Trends
- Seasonality
- War
- Unforeseen Circumstances
- Legal/Regulatory Environment

# Recommendations

- The model created for our project was different from other projects because we are predicting a return, but not recommending one specific solution
- The market is very volatile and regardless of what model is used to make investment decisions, it is important to remember that staying informed will lead to the best results
- Key Takeaway: You should be informed about the market before making investments. Our model should not be the primary factor used in making investment decisions, rather it should act as a guide in decision making

# ***Thank you!***