Final Project

2024-11-18

MIST 7770 Final Project - 9:35 Group 1

DATA PREP STEP 1

```
prices <- read_csv("/Users/bretttracy/Desktop/BI_Analytics/bi_project/prices.csv",
                   show_col_types = FALSE)
fundamentals <- read_csv("/Users/bretttracy/Desktop/BI_Analytics/bi_project/fundamentals.csv",
                         show_col_types = FALSE)

# Compute conservative and optimal return
prices <- prices %>% mutate(conservative_return = ((open-close)/open)*100,
                            optimal_return = ((high-low)/low)*100,
                            date = mdy(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),
            .groups = "drop")

# 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),
            .groups = "drop") %>%
  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"))
```

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

DATA PREP STEP 2

```r
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))

# 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))
```

```
## [1] 0
```

```r
# 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)
```

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")
```

RANDOM FOREST

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

```
##
## Call:
##  randomForest(formula = conservative_train_y ~ ., data = conservative_train,      proximity = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##           Mean of squared residuals: 0.003625949
##                     % Var explained: 16.58
```

```
# 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)
```

```r
# Print evaluation metrics
paste0("Random Forest Mean Absolute Error: ", round(mean(eval$abs_error),3))
```

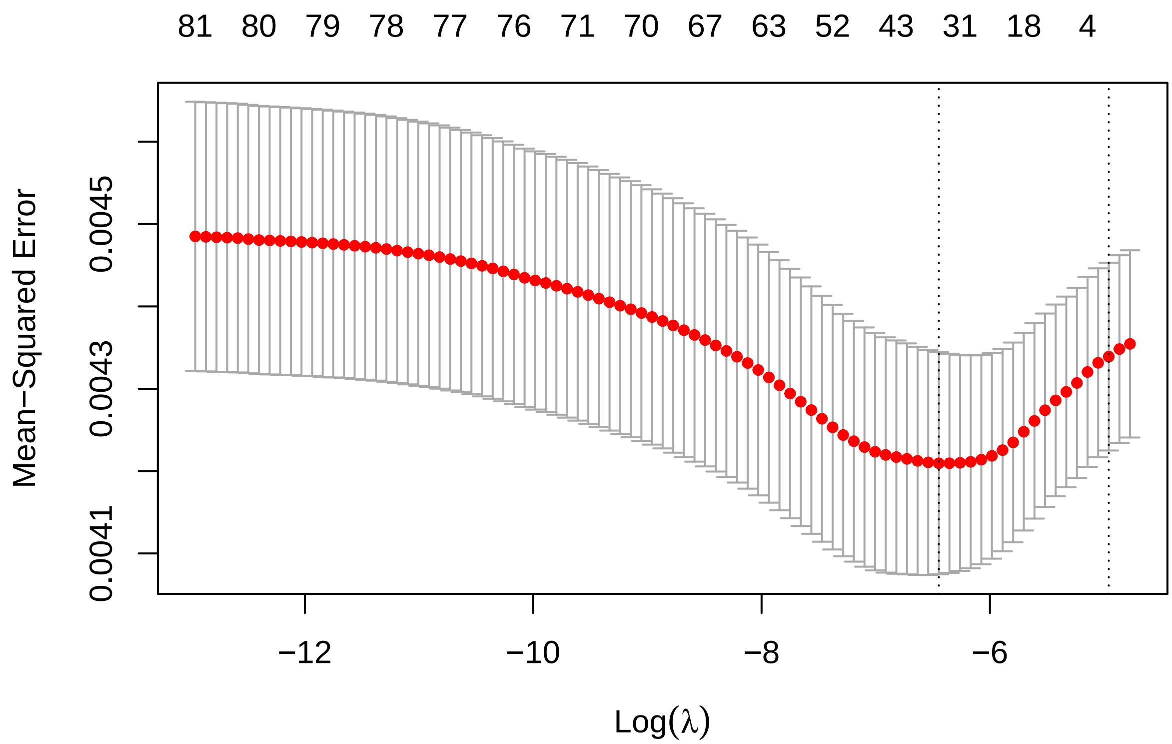
```
## [1] "Random Forest Mean Absolute Error: 0.047"
```

```r
paste0("Random Forest Mean Absolute Percentage Error: ", round(mean(eval$mape),3),"%")
```

```
## [1] "Random Forest Mean Absolute Percentage Error: 0.322%"
```

LASSO REGRESSION

```r
# 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)
```




```r
# 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)
# beta_coefficients[order(abs(beta_coefficients),decreasing=TRUE),]

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

```r
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)

paste0("Lasso Regression Mean Absolute Error: ", round(mean(eval$abs_error),3))
```

```
## [1] "Lasso Regression Mean Absolute Error: 0.052"
```

```r
paste0("Lasso Regression Mean Absolute Percentage Error: ", round(mean(eval$mape),3),"%")
```

```
## [1] "Lasso Regression Mean Absolute Percentage Error: 0.245%"
```

LINEAR MODEL

```r
# 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)
```

```
##
## Call:
## lm(formula = conservative_train_y ~ year + CashRatio + NetCashFlow +
##     OperatingMargin + MinorityInterest + ChangesinInventories +
##     AccountsReceivable + open, data = conservative_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.187623 -0.038685  0.000405  0.042739  0.187821
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.038826   0.001843 -21.063  < 2e-16 ***
## year                 -0.007266   0.001888  -3.848 0.000125 ***
## CashRatio             0.004027   0.001902   2.117 0.034449 *
## NetCashFlow           0.002274   0.001884   1.207 0.227482
## OperatingMargin       0.003091   0.001907   1.621 0.105304
## MinorityInterest      0.006968   0.001885   3.696 0.000229 ***
## ChangesinInventories -0.007887   0.001869  -4.220 2.63e-05 ***
## AccountsReceivable   -0.004797   0.001847  -2.597 0.009508 **
## open                  0.004899   0.001918   2.554 0.010762 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06436 on 1211 degrees of freedom
## Multiple R-squared:  0.05398,    Adjusted R-squared:  0.04773
## F-statistic: 8.638 on 8 and 1211 DF,  p-value: 1.64e-11
```

```r
# 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
paste0("Linear Mean Absolute Error: ", round(mean(eval$abs_error),3))
```

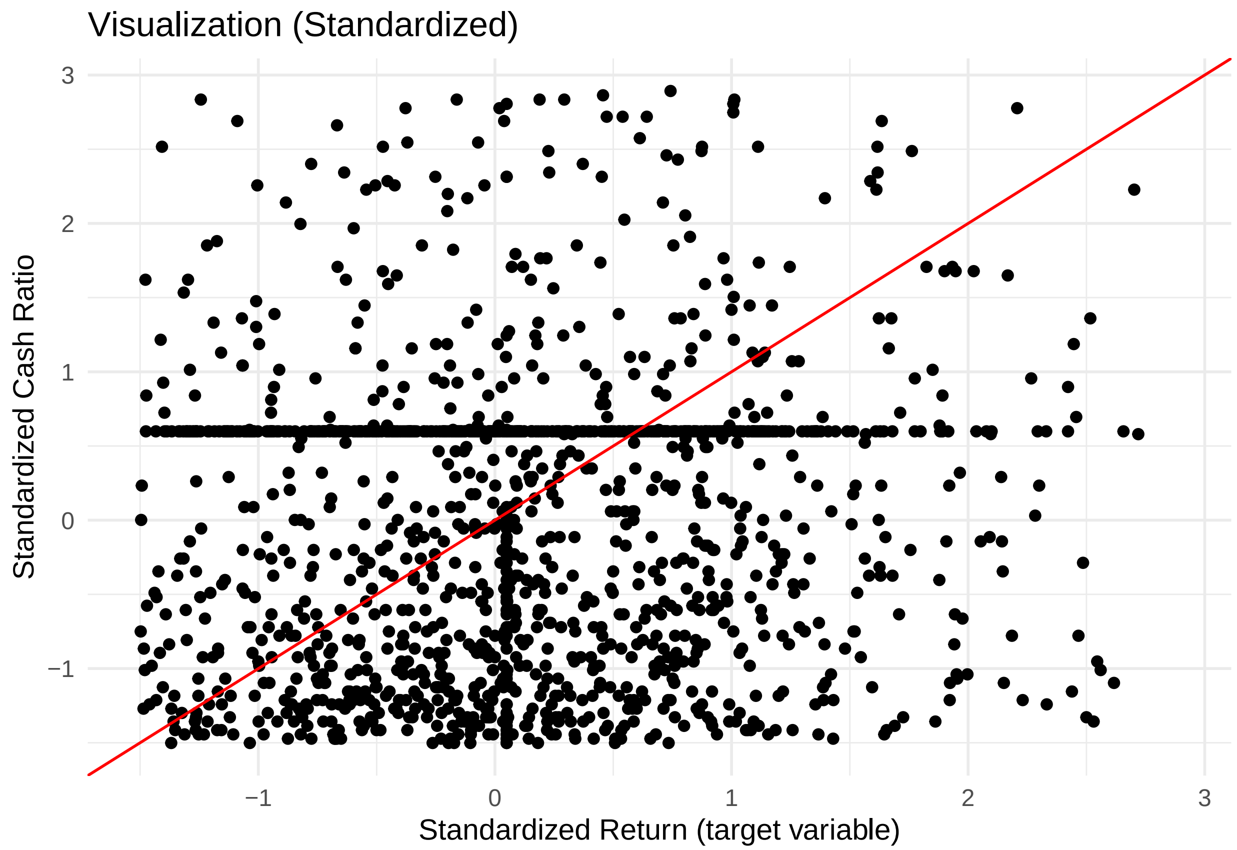
```
## [1] "Linear Mean Absolute Error: 0.052"
```

```r
paste0("Linear Mean Absolute Percentage Error: ", round(mean(eval$mape),3),"%")
```

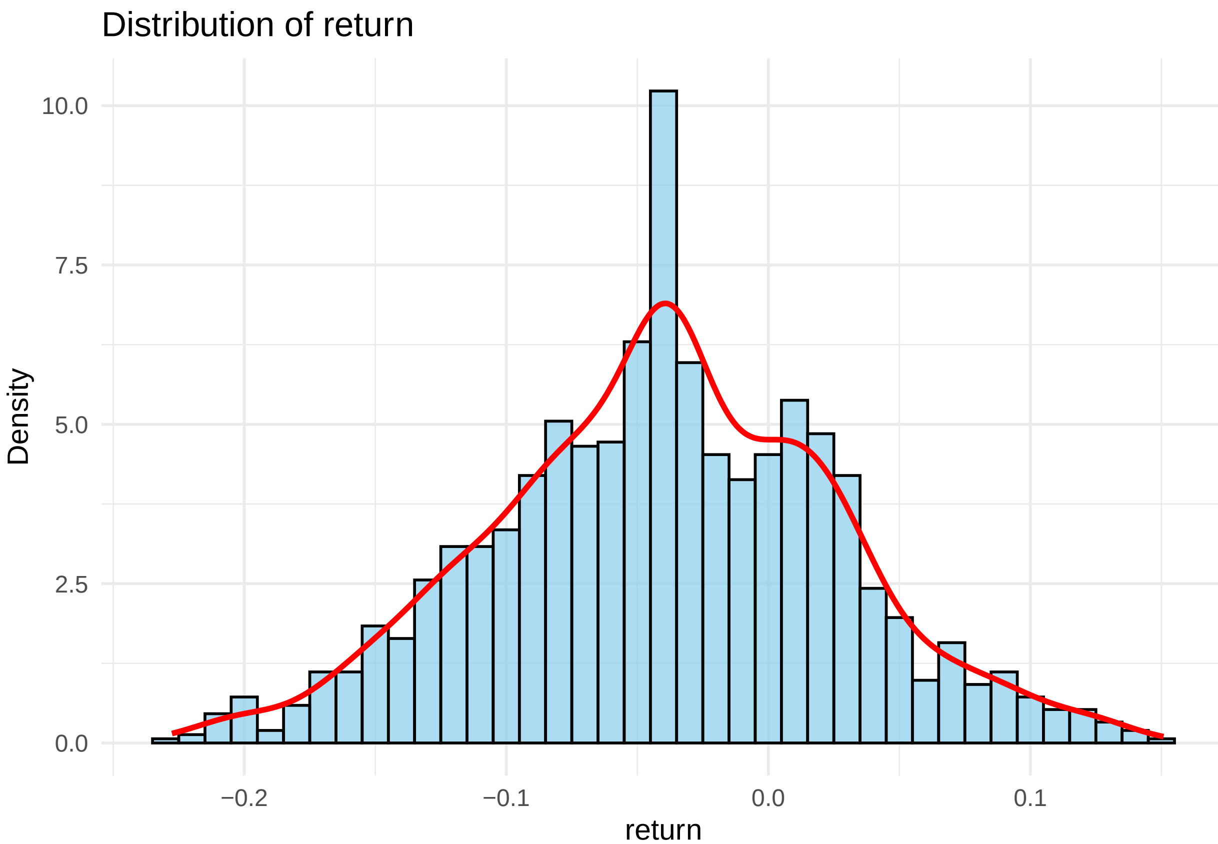
```
## [1] "Linear Mean Absolute Percentage Error: 0.183%"
```

VISUALIZATIONS

```r
# Read in prepped data
data <- read.csv("/Users/bretttracy/Desktop/BI_Analytics/bi_project/data.csv")

# Standardize data
data$meanconservativereturnscaled <- scale(data$meanconservativereturn)
data$CashRatio_scaled <- scale(data$CashRatio)

# Create range limits
range_limits <- range(c(data$mean_conservative_return_scaled, data$CashRatio_scaled), na.rm = TRUE)

# Scatter plot with Conservative Return and Cash Ratio
ggplot(data = data.frame(True = data$meanconservativereturnscaled,
                         Predicted = data$CashRatio_scaled),
       aes(x = True, y = Predicted)) + geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Visualization (Standardized)",
       x = "Standardized Return (target variable)",
       y = "Standardized Cash Ratio") +
  theme_minimal() +
  lims(x = range_limits, y = range_limits)
```




```r
# Distribution of Conservative Return
ggplot(data, aes(x = meanconservativereturn)) +
  geom_histogram(aes(y = ..density..),
                 binwidth = 0.01, fill = "skyblue", color = "black", alpha = 0.7) +
  geom_density(color = "red", size = 1) +
  labs(title = "Distribution of return",
       x = "return",
       y = "Density") +
  theme_minimal()
```


Distribution of return
10.0
7.5
5.0
2.5
0.0
Density
−0.2
−0.1
0.0
0.1
return