Stock Market Price Prediction

Exploring Machine Learning Methods in Financial Data

Aidan Kardan

Background and Objective

US Stock Market Data: **11 Datasets** including AMZN stock data and various predictors, spanning fundamental, technical, and raw historical metrics

Analysis Goals:

- **Regression Task**: Predict future AMZN stock prices
- **Classification Task**: Predict stock movement (up/down) based on past information

TABLE OF CONTENTS

Data Cleaning

First, I had to merge 11 datasets into one to set up the statistical analysis

Second, I had to deal with missing values and problems that arose from merging several datasets together

For financial data, missing values need to be handled carefully

Missing Values Handling:

- Forward-Fill Financial Ratios
- Forward-Fill Miscellaneous NA established on rare occasions where macroeconomic data is missing
- Remove Dates with no open, high, low, close, and volume data as it indicates no trading activity occurred

Data Description

- In my analysis, the training data spans **January 2010 to December 2019** and the test data spans **January 2020 to July 2022**

- The variable of interest for prediction, i.e. the response variable, is **Close price**

- The response variable price range for entire data: **[5.43,186.5]**

- There are **97 available predictors** in the dataset and **3,135 observations** for each predictor

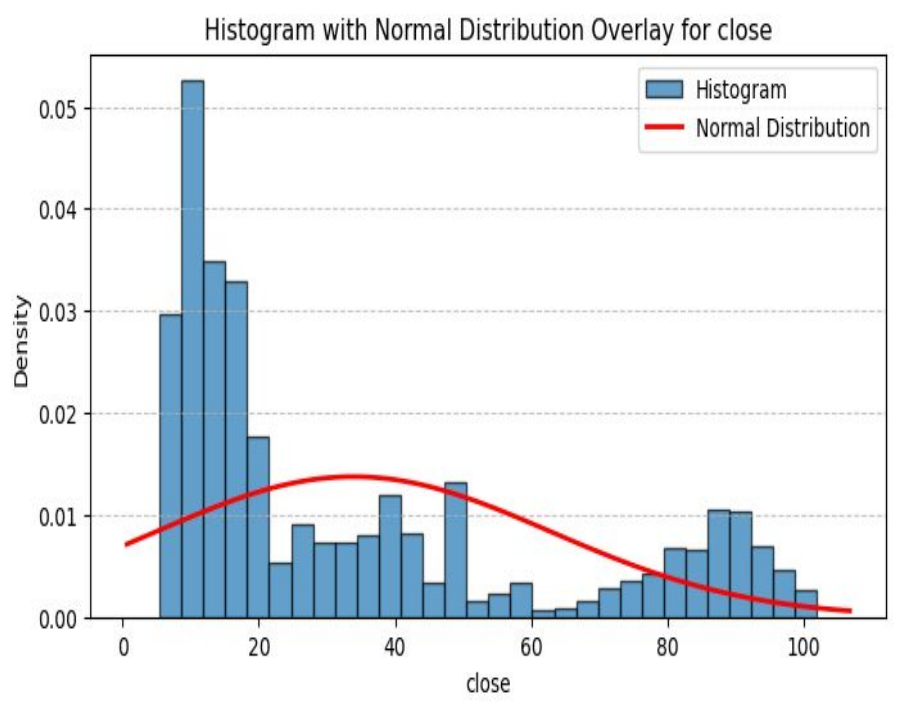
Time Series Data Issues

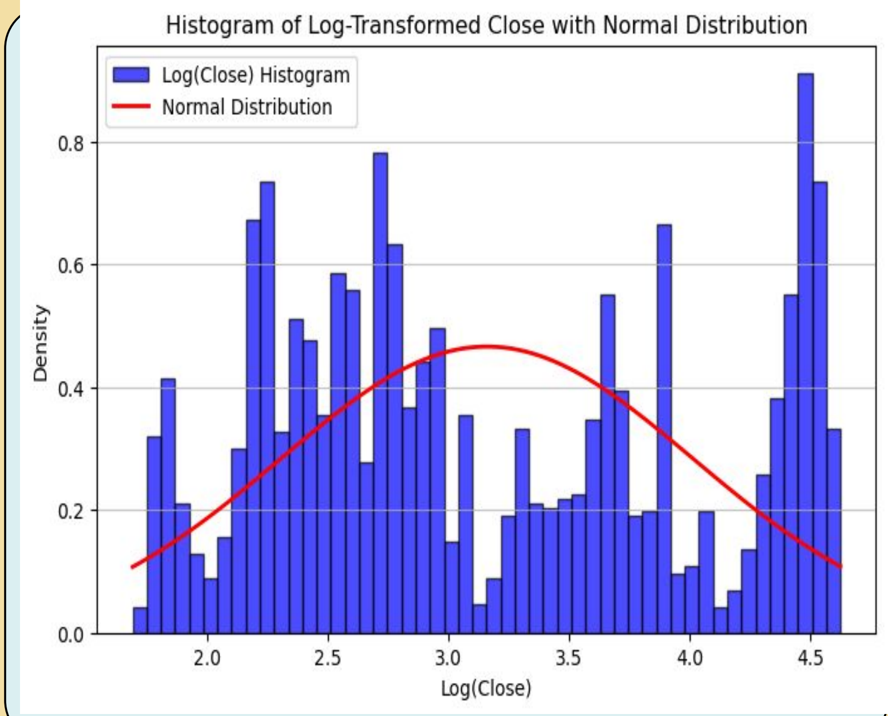
When dealing with time series data, standard training-test splits with randomization, and Cross-Validation or Bootstrap techniques cannot be used for analysis

To fix this, we will utilize a **time-series split/time-series CV**, which preserves the integrity of the chronological order of observations by sequentially splitting the data

This method ensures that future data points are never used to predict past ones, maintaining the temporal structure critical for accurate financial modeling

Exploratory Analysis: Histograms

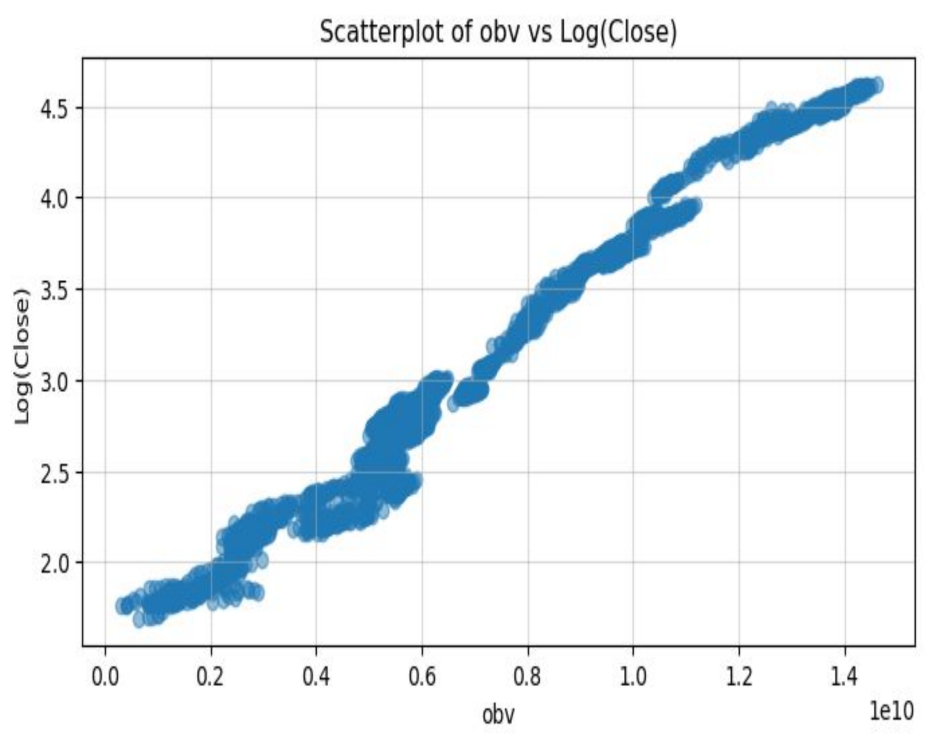




Exploratory Analysis: Scatterplots

Scatterplots of each predictor with the transformed response help visualize and diagnose linear or non-linear relationships between the response and the predictor

On Balance Volume: TA Indicator measuring cumulative flow of trading volume by adding volume on up days and subtracting volume on down days

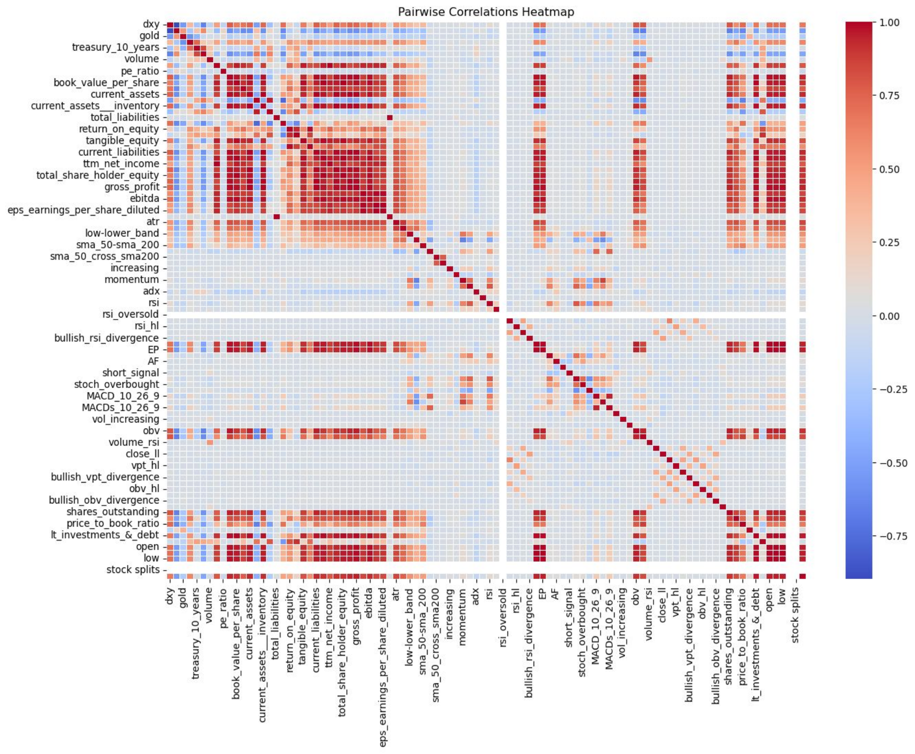


Exploratory Analysis: Correlation

Correlation Analysis:

- Pairwise Correlations
- Correlation of each predictor to transformed response, log close

The main takeaway from the heatmap is that **financial ratios predictors** are highly correlated with each other and **technical analysis based predictors** are highly correlated with each other



Exploratory Analysis: Correlation

After calculating the correlation of each variable to the transformed response, log close, there are several potential steps

Initial Correlation Filtering to reduce Dimensionality:

It is standard practice to include variables with at least |0.3| correlation to the response

Initial Correlation Filtering leaves me with **44 predictors**

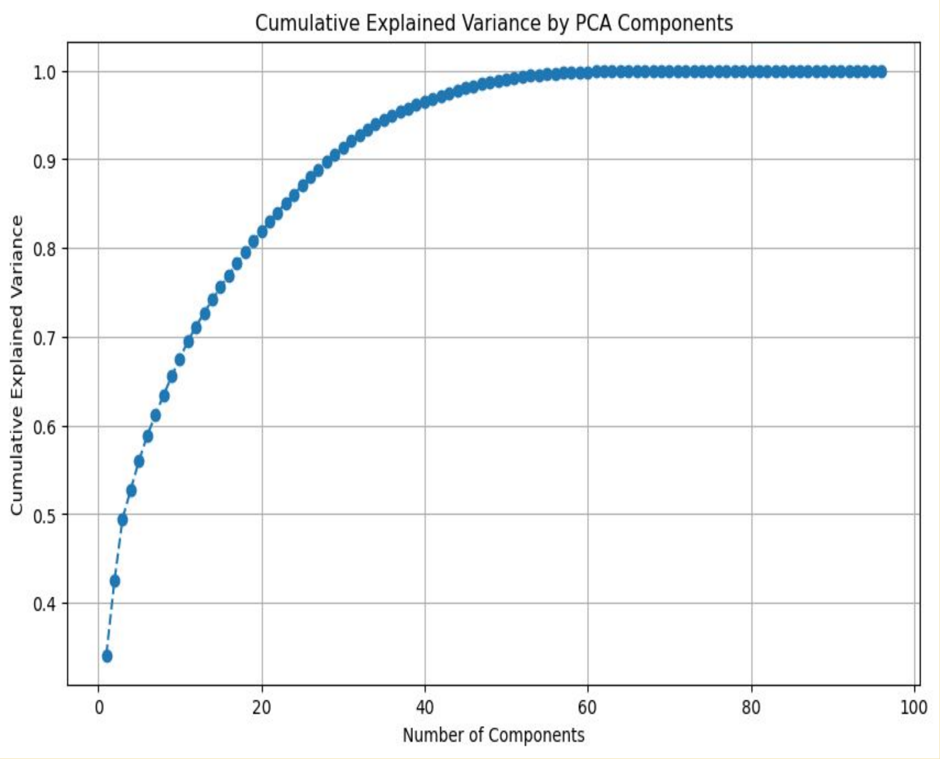
We will now explore further methods to reduce dimensionality and help us **select our variables of interest** for our predictive model

Feature Importance: PCA

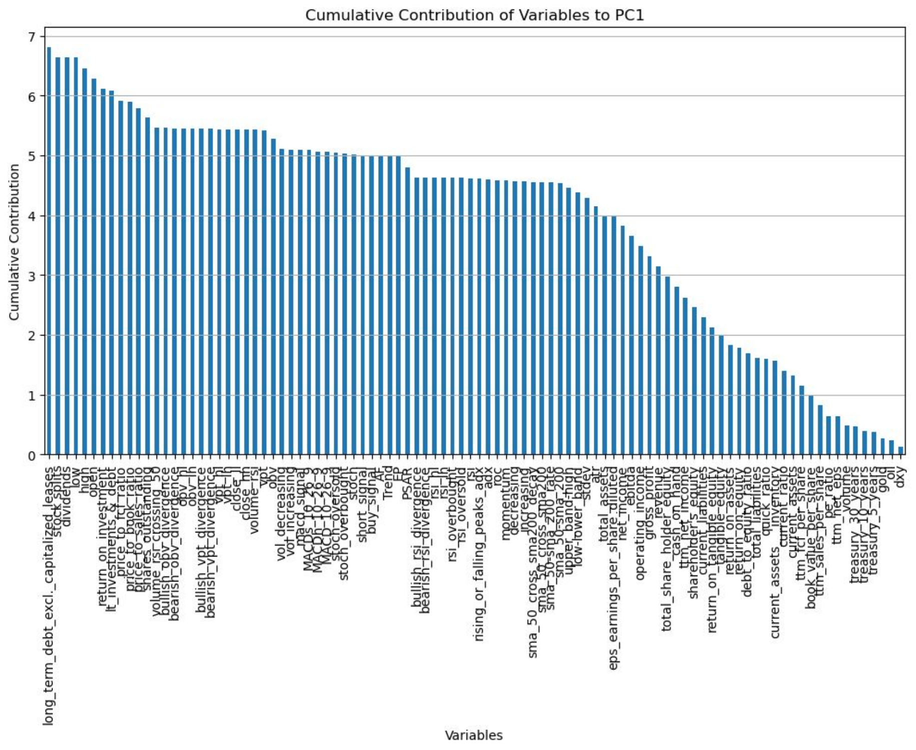
Top 3 PC Components:

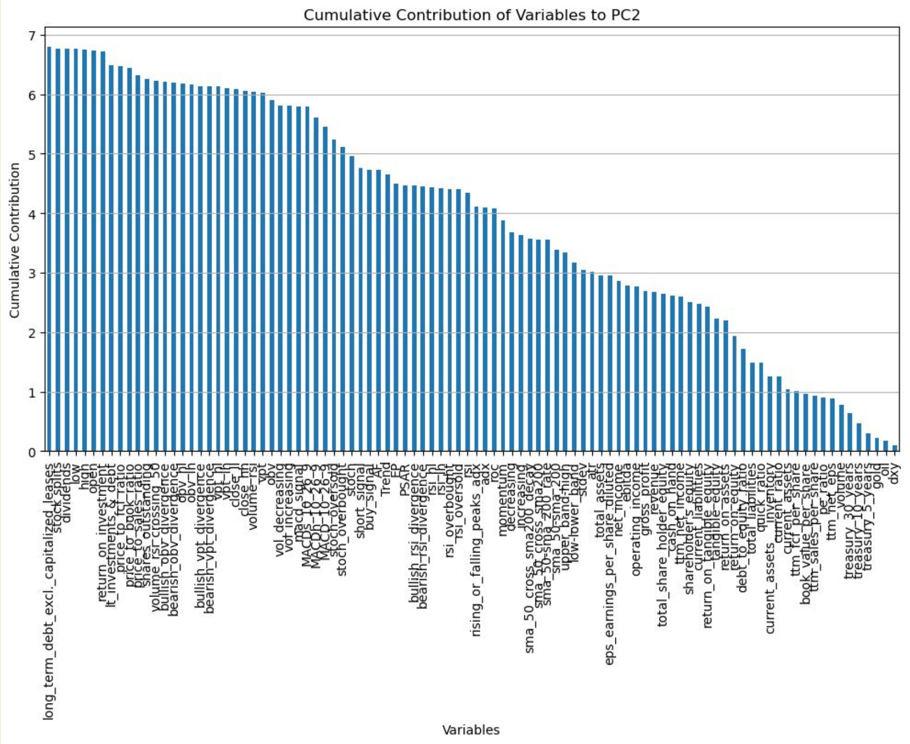
- PC1: **34.1%** Explained Variance
- PC2: **8.37%** Explained Variance
- PC3**: 7.03%** Explained Variance

In order to explain over **95%** of the variance we need to consider more than 40 principle components

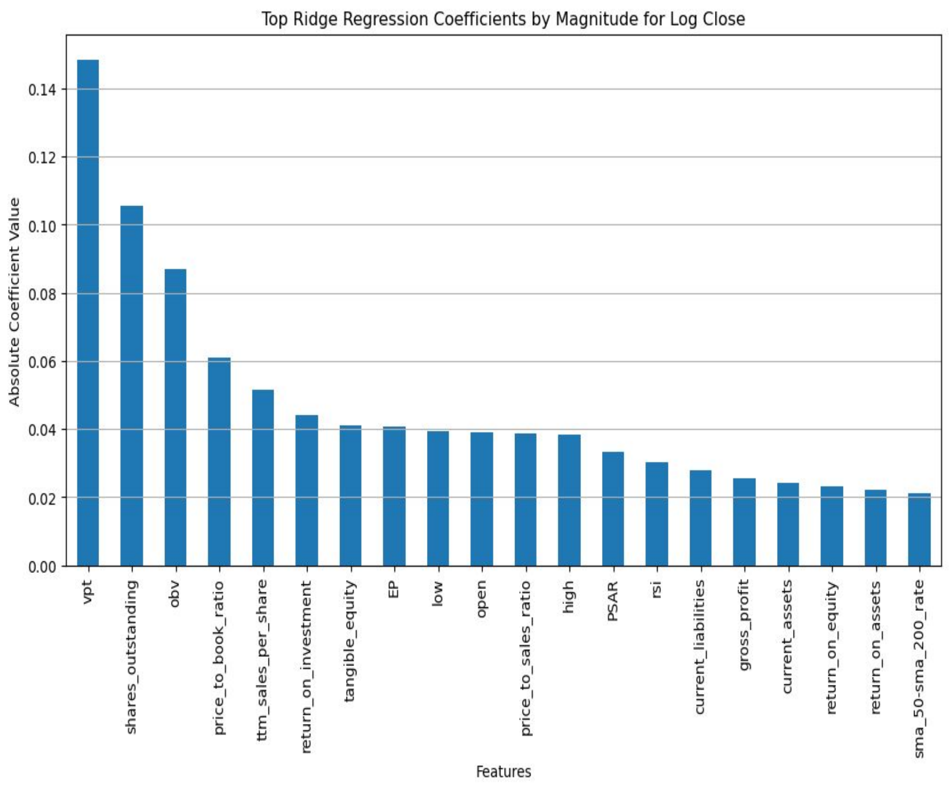


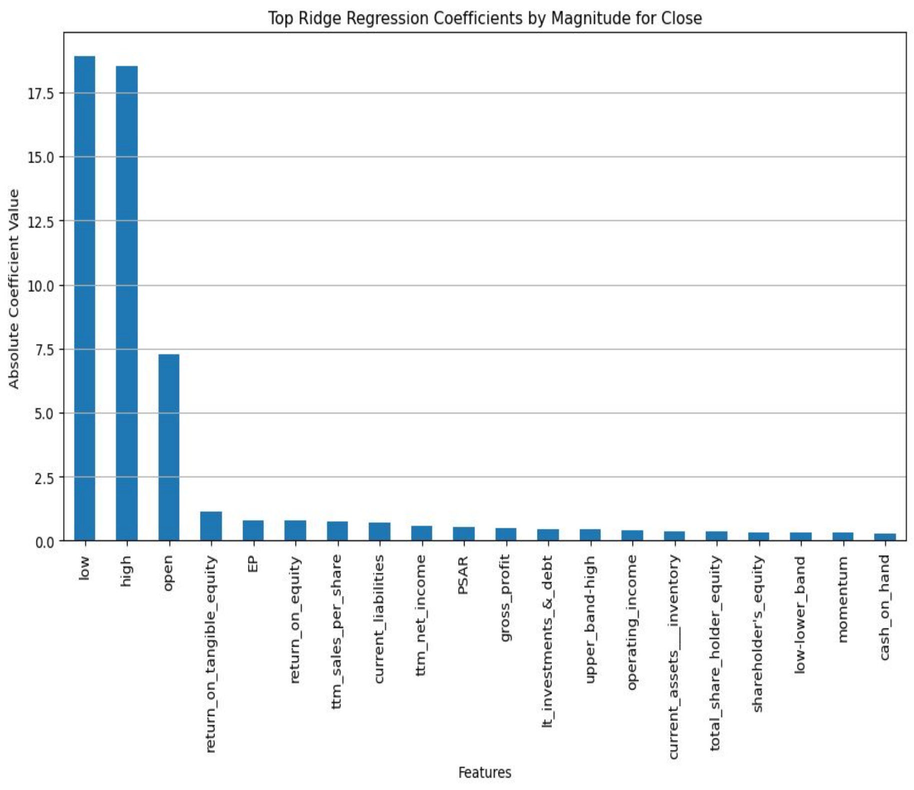
Feature Importance: PCA



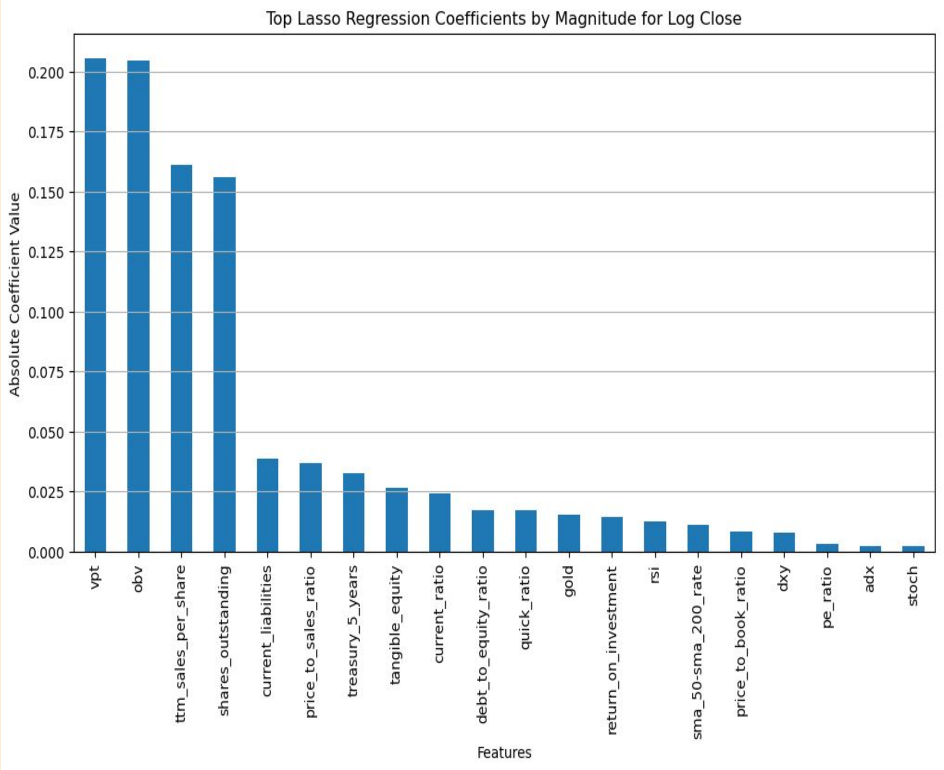


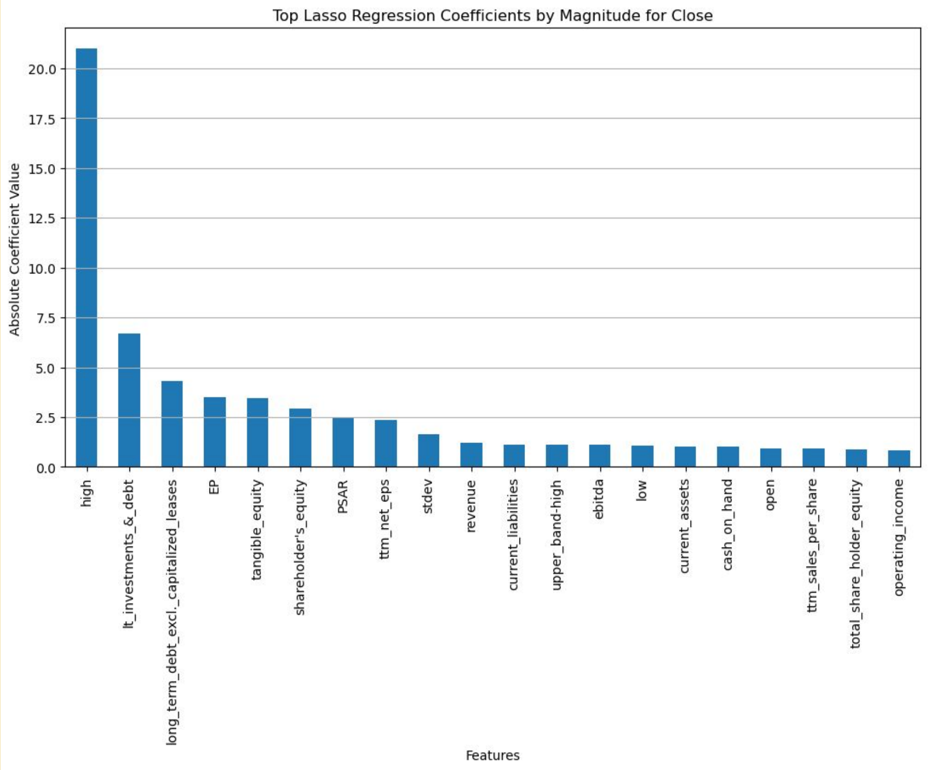
Feature Importance: Ridge Regression



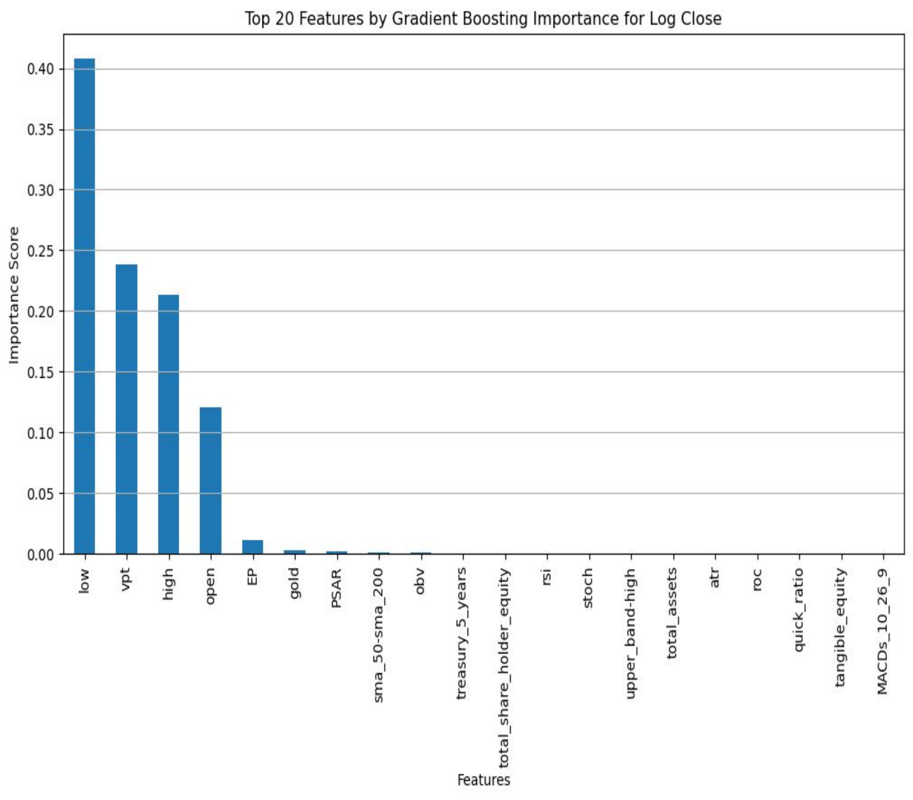


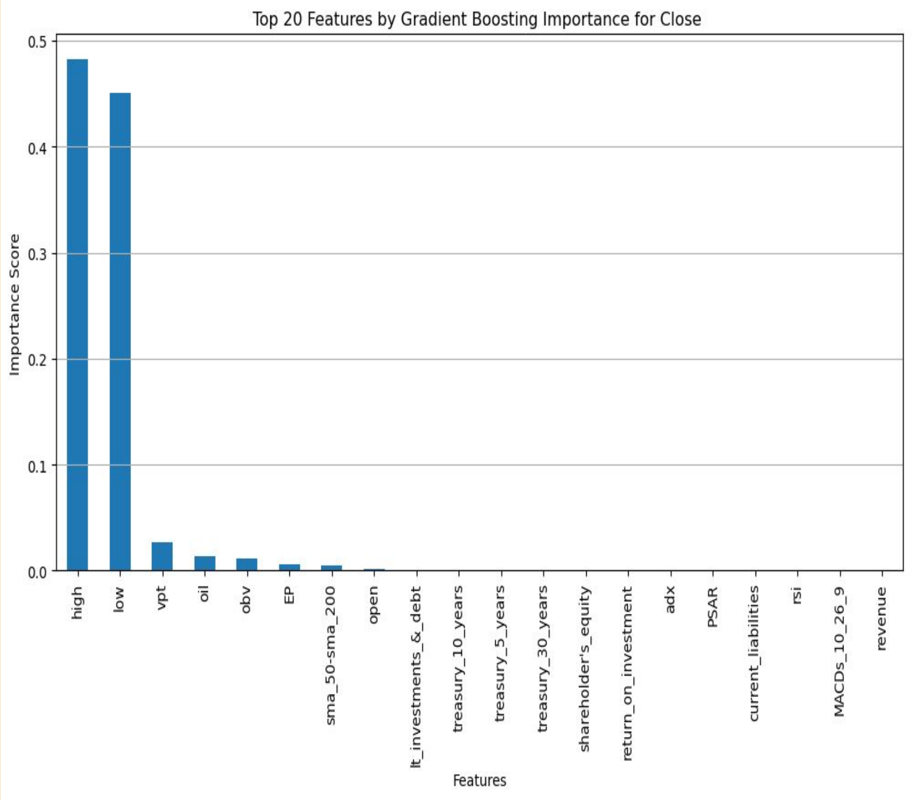
Feature Importance: Lasso Regression



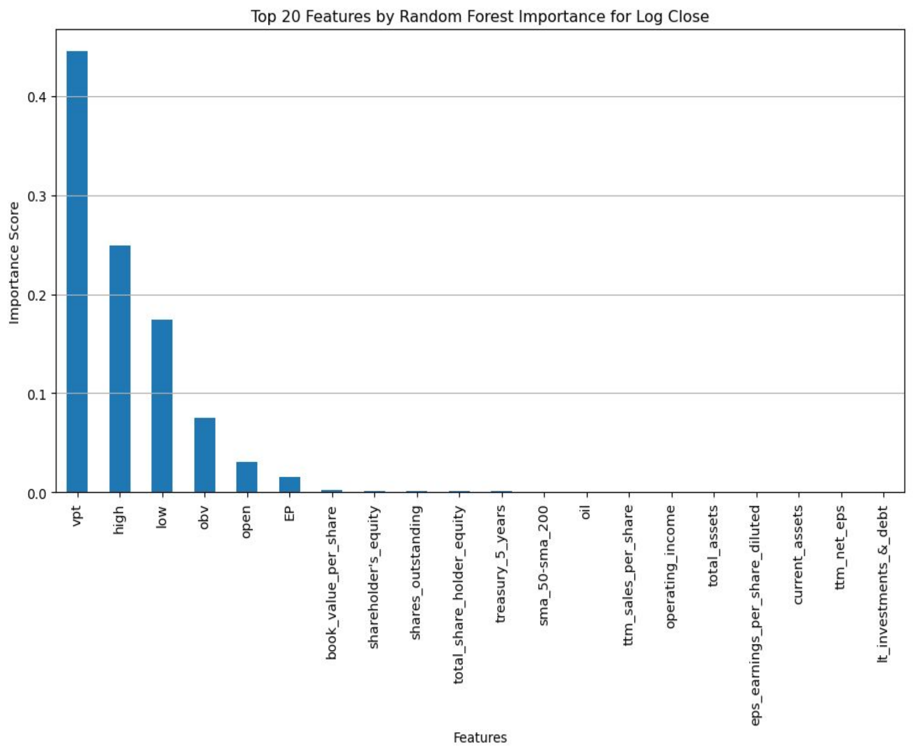


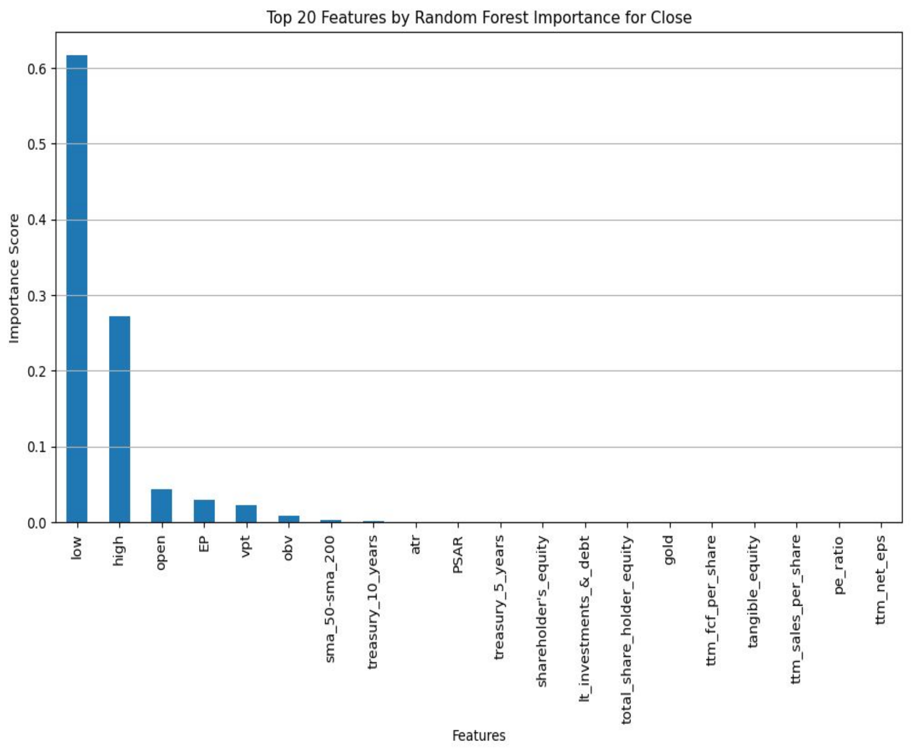
Feature Importance: Gradient Boosting



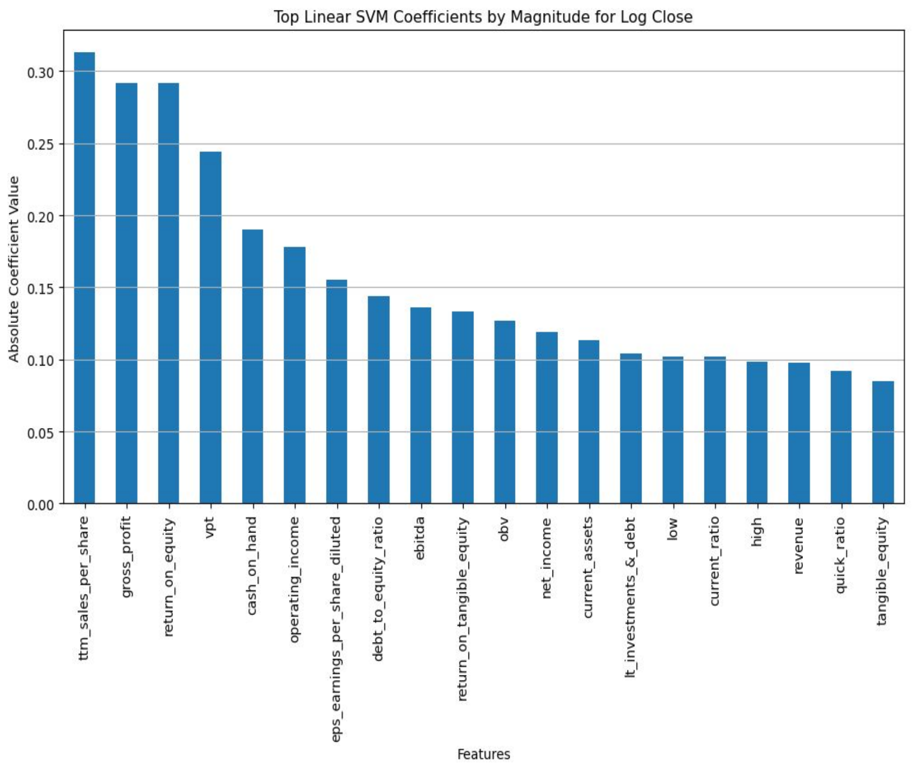


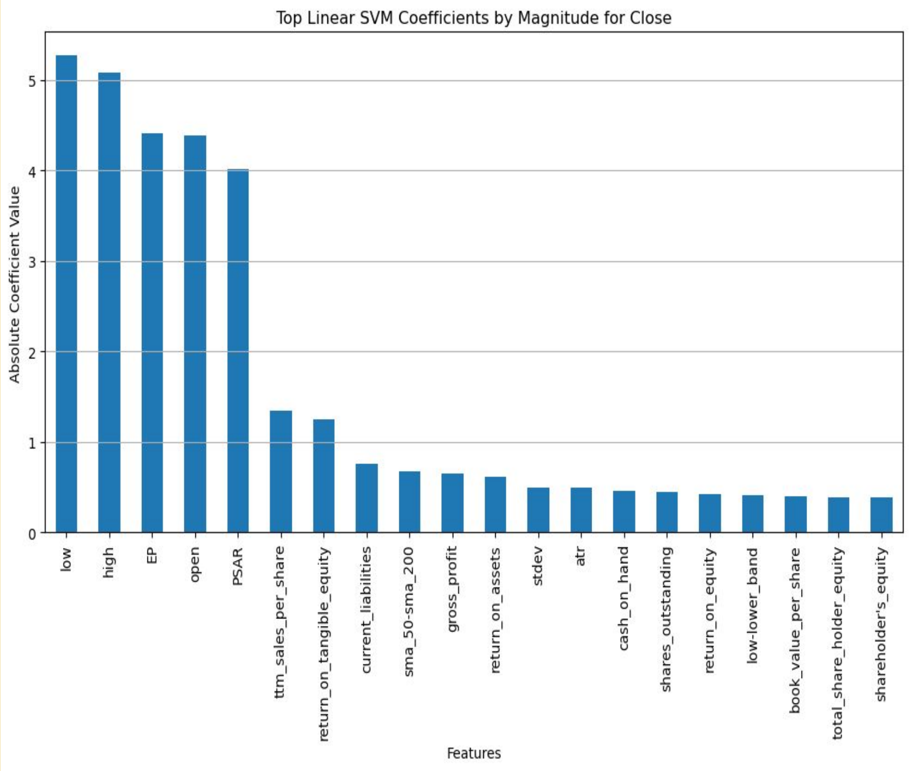
Feature Importance: Random Forests



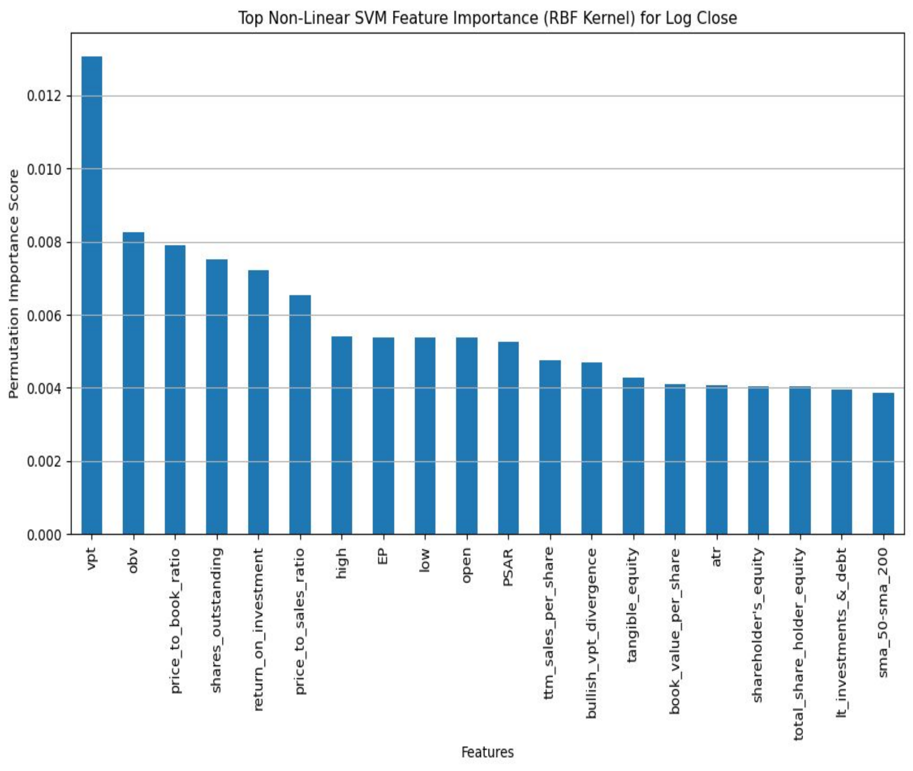


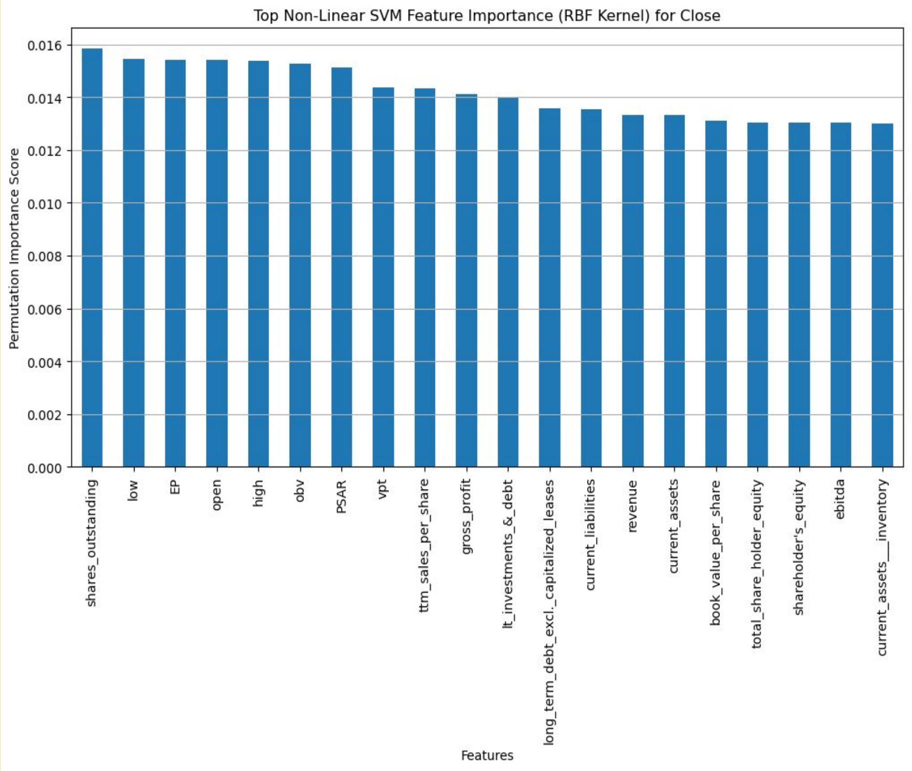
Feature Importance: Linear SVM





Feature Importance: Non-Linear SVM

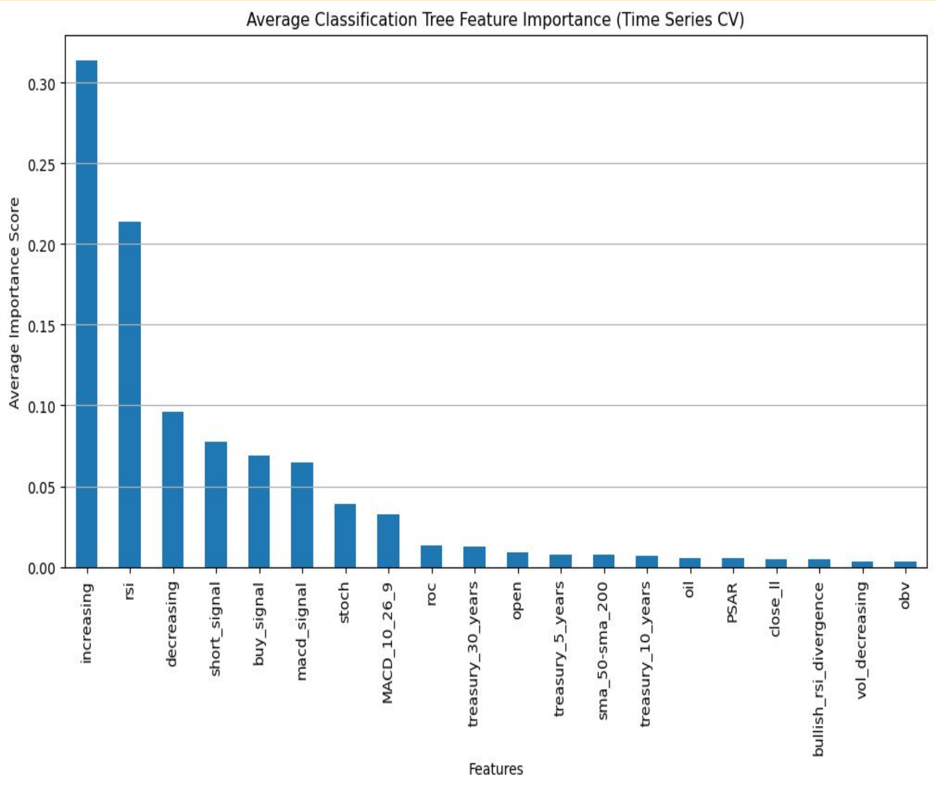




Feature Importance: Classification Trees

In order to do this, I had to create a variable called **Direction**, which was a new **binary variable** taking 1 when the close price increased from its previous close price and 0 otherwise

Later analysis will further explore the classification problem



KMeans Clustering

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt

# Standardize predictors
scaler = StandardScaler()
X_scaled = scaler.fit_transform(numeric_data)

# Perform K-Means clustering
kmeans = KMeans(n_clusters=5, random_state=42, algorithm="elkan")
kmeans_clusters = kmeans.fit_predict(X_scaled)

# Ensure 'Date' is in datetime format
train_data1 = train_data.copy()  # Create a copy to avoid
SettingWithCopyWarning
train_data1.loc[:, 'KMeans_Cluster'] = kmeans_clusters
train_data1.loc[:, 'Date'] = pd.to_datetime(train_data['Date'])  # Ensure
Date is in datetime format

# Summarize response ('close') by cluster
kmeans_summary = train_data1.groupby('KMeans_Cluster').mean()
 # Visualize cluster distribution over time
 plt.figure(figsize=(12, 6))
 for cluster in sorted(train_data1['KMeans_Cluster'].unique()):
     cluster_dates = train_data1[train_data1['KMeans_Cluster'] ==
 cluster]['Date']
     plt.scatter(cluster_dates, [cluster] * len(cluster_dates),
 label=f"Cluster {cluster}", alpha=0.6)
 plt.title("K-Means Clusters Over Time")
 plt.xlabel("Date")
 plt.ylabel("Cluster")
 plt.legend(title="Clusters")
 plt.grid(axis='y')
 plt.show()
```

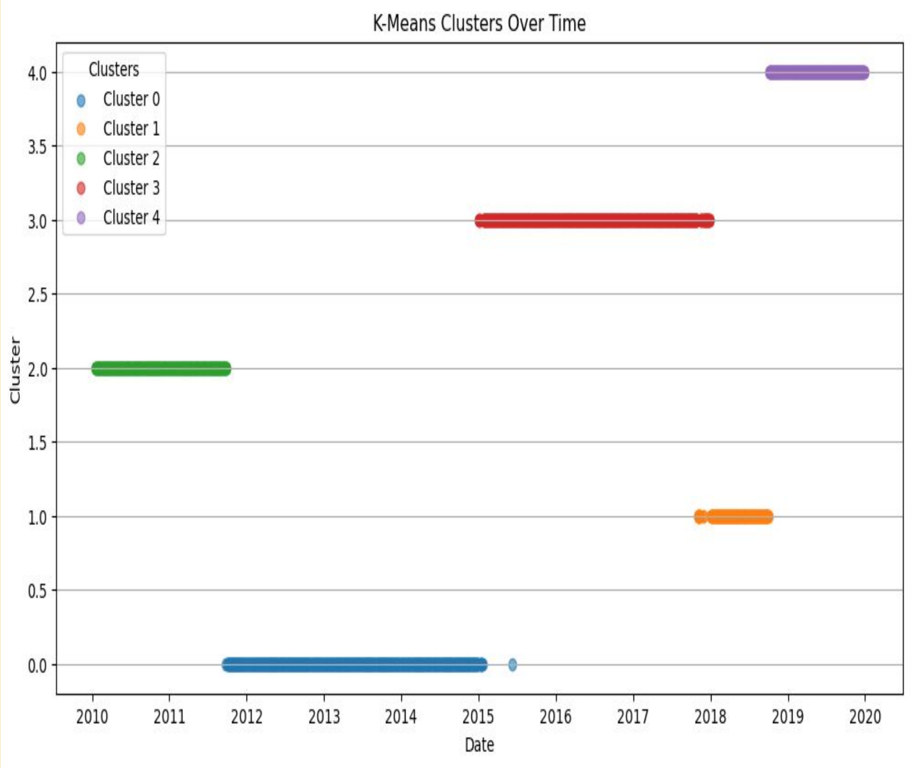
0: 13.89

1: 80.50

2: 8.14

3: 35.65

4: 88.09



Model-Based (GMM) Clustering

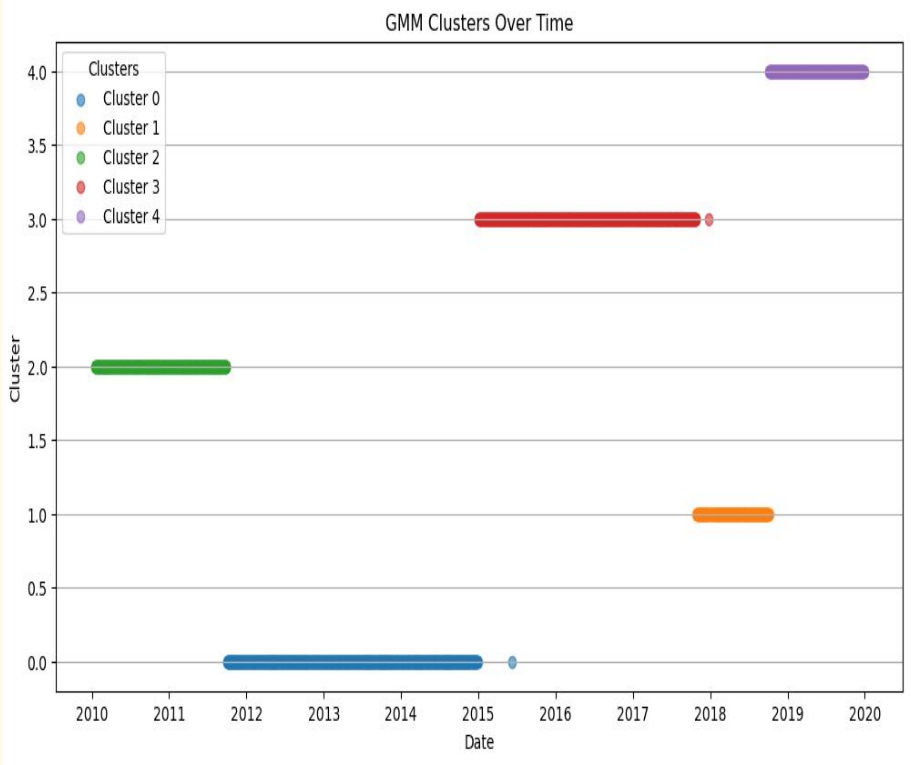
```
from sklearn.mixture import GaussianMixture
import pandas as pd
import matplotlib.pyplot as plt

# Perform Gaussian Mixture Clustering
gmm = GaussianMixture(n_components=5,
random_state=42)
gmm_clusters = gmm.fit_predict(X_scaled)

# Add GMM cluster labels to the dataset
train_data3 = train_data.copy()
train_data3.loc[:, 'GMM_Cluster'] = gmm_clusters

# Visualize GMM clusters over time
plt.figure(figsize=(12, 6))
for cluster in sorted(train_data3['GMM_Cluster'].unique()):
    cluster_dates =
train_data3[train_data3['GMM_Cluster'] == cluster]['Date']
    plt.scatter(cluster_dates, [cluster] * len(cluster_dates),
label=f"Cluster {cluster}", alpha=0.6)
plt.title("GMM Clusters Over Time")
plt.xlabel("Date")
plt.ylabel("Cluster")
plt.legend(title="Clusters")
plt.grid(axis='y')
 plt.show()
```

0: 13.89
1: 77.59
2: 8.15
3: 34.88
4: 88.07



Summary of Feature Importance

Across the **7 different methods** with a log transformed response, when considering the top 10 most important features (70 total) there were **39 unique features**

Now we will evaluate each model on its own **using time-series CV** and extract **RMSE** and compare them

The goal is to create an all-encompassing model looking to improve on the best baseline model

RMSE Calculations

For each model's RMSE, we evaluated performance by fitting on a training set and testing on a validation set using **time-series CV**

This approach ensured the models were trained and assessed in alignment with the chronological structure of the data

In all baseline model calculations, I used **all predictors and the log transformed response**

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error
import pandas as pd
import matplotlib.pyplot as plt

# Prepare predictors and response
X = numeric_data.drop(columns=['close', 'log_close'], errors='ignore')  # Exclude response
y = train_data['log_close']  # Log-transformed response variable

# Time series split
tscv = TimeSeriesSplit(n_splits=5)

# Initialize list to store RMSE  for each split
rmse_list = []

# Iterate through time series splits and fit Gradient Boosting
for train_index, test_index in tscv.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Train Gradient Boosting
    gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)
    gb_model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = gb_model.predict(X_test)

    # Calculate RMSE
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    rmse_list.append(rmse)
```

Summary of RMSE for Baseline Models

Linear Regression Average RMSE (Original Scale): **Inf**
Ridge Regression Average RMSE (Original Scale): **1.047094313711389**
Lasso Regression Average RMSE (Original Scale): **1.0424849102056157**
PCA Regression Average RMSE (Original Scale): **1.424176664354599**
Random Forest Average RMSE (Original Scale): **1.2721996992990103**
Gradient Boosting Average RMSE (Original Scale): **1.2756891464391382**

Linear SVM Average RMSE (Original Scale): **1.5948725779569104**
Non-linear SVM Average RMSE (Original Scale): **1.8346066104355785**
KMeans Clustering RMSE for Log Close (Original Scale): **1.2687367660491O3**
Hierarchical Clustering RMSE for Log Close (Original Scale): **1.451092694116358 2**
Model-Based Clustering (GMM) RMSE for Log Close (Original Scale): **1.2692909733396893**
Classification Tree Accuracy: **0.61**

Normalized RMSE for Baseline Models

Linear Regression: **Inf**

Ridge Regression: **0.005782815009175397**

Lasso Regression: **0.005757358536508619**

PCA Regression: **0.007865337517836l9**

Random Forest: **0.007026010378853544**

Gradient Boosting: **0.007045281639361232**

Linear SVM: **0.008808044280979237**

Non-linear SVM: **0.010132029659444295**

KMeans Clustering: **0.007006885547297195**

Hierarchical Clustering: **0.008013987375690939**

Model-Based (GMM) Clustering: **0.007009946282320038**

Interpretation and RMSE Reasoning

Normalized RMSE: x = RMSE / range of response

Normalized RMSE tells me that the model's error is**, on average, about x% of the total range of the response variable**

All normalized RMSE for Baseline Models are **below 1.02%** meaning the models are capturing a significant portion of the variability in the data and demonstrate **great predictive power**

RMSE was chosen because I am predicting close prices, and my focus is on the **absolute difference** between predictions and actual observations

RMSE is expressed in the **same units** as the close price, making it both intuitive and easy to understand

The best model was Lasso considering all predictors

Optimal Mixed Model

Based on Feature Importance Analysis and the Baseline Models considering all predictors RMSE:

- The next step is to focus on the **39 unique features** identified from the baseline models
- I will also incorporate the best-performing clustering method **(KMeans)** as a categorical variable to capture potential non-linear relationships in the data

Calculate RMSE on Linear, PCA, Ridge, and Lasso Regression, Random Forest, Gradient Boosting, Linear SVM for **40 predictor model** and compare to baseline models with all predictors

Summary of RMSE for Optimally Selected Model

Linear Regression RMSE (Original Scale): **1.3166661668616408**
Ridge RMSE (Original Scale): **1.2355178856459195**
Lasso RMSE (Original Scale): **1.2071336836237347**
PCA Regression RMSE (Original Scale): **1.2641572788408475**

Random Forest RMSE (Original Scale): **1.2698637777706667**

Gradient Boosting RMSE (Original Scale): **1.2753331882329946**

Linear SVM RMSE (Original Scale): **1.5272381690448305**

Advantages and Downfalls of Chosen Model

A **baseline regression model** using 40 selected predictors performed **extremely well** compared to the full model regression

Ridge and Lasso: These methods, designed to handle multicollinearity, **performed worse** when predictors were removed. The reduction in predictive power outweighed their regularization benefits

Random Forest, Gradient Boosting, PCA Regression, Linear SVM: These models showed **improved performance** when irrelevant predictors were removed. The feature selection process helped these models focus on the most important predictors

Summary, Next Steps

1

Lasso trained on time-series CV considering all predictors performed better than any model with selected predictors based on exploratory analysis and feature importance **likely due to the complexity of financial data**

2

Next, I will explore TSA to better **capture lagged relationships, address non-stationarity, and preserve the temporal dependencies** inherent in financial data.
The goal is to identify **key lagged terms or patterns and add them as predictors** to the models to improve predictive performance

3

I plan to investigate **Neural Networks (NNs)** and **Long Short-Term Memory (LSTM)** networks due to their proven ability to handle complex, non-linear patterns and sequential data effectively as well as further explore the classification problem


WKN
BID
ASK
AKTUELL
%
514000 21 17,750 € 17,751 € 17,750 € 2,90 %
555200 6 30,862 € 30,865 € 30,864 € 1,73 %
659990 1 95,212 € 95,236 € 95,224 € 1,53 %
766403 29 128,891 € 128,930 € 128,910 € 1,46 %
A1ML7J 4 29,991 € 29,997 € 29,994 € 1,26 %
15,840 € 15,841 € 15,841 € 1,25 %
04.10.
01.11.
01.12.
KONFIGURATION
LS - DAX - Lang & Schwarz 0,79%
ZEITEINHEIT
ZEITRAUM
INDIKATOR
SCANS
LS - DAX (846900)
5 Min

THANK YOU