DTSA_5510_Final

February 21, 2025

1 DTSA 5510 Final Project

1.1 Introduction

This notebook presents a comprehensive analysis of daily fluctuations and trends within the S&P 500 index, a crucial barometer of the US stock market. Our investigation aims to explore the potential of machine learning in understanding and predicting short-term movements in this key index. We begin by visualizing historical stock data of prominent S&P 500 constituents, gaining insights into the distribution of daily percentage changes and identifying potential relationships between individual stock performance and the overall index. This exploratory data analysis forms the foundation for our core objective: developing a predictive model to forecast increases in the S&P 500 index.

1.2 Set up

1.2.1 Project Setup, Baseline Analysis, and Initial Data Exploration

This section lays the groundwork for our analysis by establishing the project's structure, creating a baseline for comparison, and conducting an initial exploration of the raw data. We begin by importing necessary libraries, including those for data manipulation (pandas, numpy), visualization (matplotlib, seaborn), and machine learning (scikit-learn). This ensures we have the required tools for data processing, analysis, and model building. We also define key project parameters, such as the timeframe for our historical data and the specific S&P 500 constituents we will be analyzing. This sets the scope of our investigation and ensures reproducibility of our results.

A crucial first step is acquiring the raw data. We describe the data source and the method for retrieving historical stock price information for our chosen S&P 500 companies. This data typically includes daily open, high, low, and closing prices, as well as trading volume. We emphasize the importance of data integrity and discuss any potential limitations or biases in the data source. This transparency is essential for ensuring the reliability of our subsequent analysis and modeling.

Before diving into complex analysis, we establish a simple baseline for comparison. This baseline provides a benchmark against which we can evaluate the performance of our more sophisticated machine learning models. A common baseline in financial forecasting could be a naive prediction, such as assuming the S&P 500 index will move in the same direction as it did the previous day. Alternatively, we might calculate simple moving averages or other basic indicators to serve as a rudimentary forecasting method. Establishing this baseline helps us understand the inherent difficulty of the prediction task and provides context for evaluating the added value of our machine learning approach.

The initial data exploration focuses on understanding the raw data's characteristics. We use descriptive statistics (mean, median, standard deviation, etc.) to summarize the distribution of stock prices and trading volume. We examine the time series of stock prices visually, looking for any obvious trends, seasonality, or unusual patterns. This initial exploration helps us identify potential issues with the data, such as missing values or outliers, which might require further preprocessing. We also begin to explore the relationships between different stocks, perhaps by calculating correlation coefficients between their daily returns. This preliminary analysis provides valuable context for our subsequent feature engineering and model building efforts. By carefully examining the raw data, we gain a deeper understanding of the underlying patterns and dynamics of the stock market, which informs our choices in the later stages of the project.

1.2.2 Inport needed tools

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
  ↪classification_report
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

1.2.3 Data setup

[2]:
```
df= pd.read_csv('sp500_trends.csv')
df
```

[2]:
```
      Unnamed: 0        Date  sp500_increase  sp500_changep  ADBE_increase  \
0              0  2021-04-19               0      -0.252251              0
1              1  2021-04-16               1       0.002399              1
2              2  2021-04-15               1       0.514817              1
3              3  2021-04-14               0      -0.334272              0
4              4  2021-04-13               1       0.323181              1
…            …          …               …             …             …
2123        2123  2012-05-24               0      -0.075402              0
2124        2124  2012-05-23               1       0.777146              1
2125        2125  2012-05-22               0      -0.083139              1
2126        2126  2012-05-21               1       1.390594              1
2127        2127  2012-05-18               0      -1.240762              0

      ADBE_changep  INTC_increase  INTC_changep  MSFT_increase  MSFT_changep  \
0        -1.330457              0     -1.653780              0     -0.557290
1         0.165963              0     -0.887803              1      0.489455
```

```
2         1.469936              1      1.641387             1      0.608695
3        -0.655641              0     -1.714891             0     -0.734043
4         1.580345              0     -0.594421             1      0.478108
…            …            …            …            …            …
2123     -1.958338              1      0.312866             0     -0.308643
2124      1.514194              0     -0.117781             0     -0.817716
2125      0.125114              0     -0.913589             1      0.235769
2126      1.780596              1      0.345357             1      2.233676
2127     -2.461056              0     -0.647866             0     -1.745554

      …  DIS_increase  DIS_changep  NFLX_increase  NFLX_changep  \
0     …             0    -0.042665              1      1.378676
1     …             0    -0.165278              0     -0.726559
2     …             0    -1.148388              1      0.928017
3     …             1     0.801207              0     -2.676298
4     …             0    -0.053877              0     -0.587077
…   …           …            …            …            …
2123  …             1     0.067550              0     -1.747768
2124  …             0    -0.067818              1      6.580706
2125  …             0    -0.224775              0     -5.696382
2126  …             1     1.323894              1      2.705798
2127  …             0    -1.461983              0     -3.236508

      TSLA_increase  TSLA_changep  META_increase  META_changep  KFC_increase  \
0                 0     -0.690660              0     -0.908173             1
1                 1      1.527485              0     -0.645754             1
2                 0     -0.571924              1      0.483127             1
3                 0     -4.991566              0     -1.457853             1
4                 1      6.962260              0     -0.784722             1
…            …            …            …            …            …
2123              0     -3.103970              1      0.242786            -1
2124              1      1.505255              1      2.008285            -1
2125              1      2.325555              0     -4.937139            -1
2126              1      4.314699              0     -6.843690            -1
2127              0     -2.855130              0     -9.084421            -1

      KFC_changep
0        0.096339
1        0.241546
2        0.096712
3        0.241663
4        0.828057
…           …
2123     0.000000
2124     0.000000
2125     0.000000
2126     0.000000
```

```
2127     0.000000

[2128 rows x 24 columns]
```

[3]:
```
df= df.fillna(0)
df.head()
```

[3]:
```
   Unnamed: 0        Date  sp500_increase  sp500_changep  ADBE_increase  \
0           0  2021-04-19               0      -0.252251              0
1           1  2021-04-16               1       0.002399              1
2           2  2021-04-15               1       0.514817              1
3           3  2021-04-14               0      -0.334272              0
4           4  2021-04-13               1       0.323181              1

   ADBE_changep  INTC_increase  INTC_changep  MSFT_increase  MSFT_changep  \
0     -1.330457              0     -1.653780              0     -0.557290
1      0.165963              0     -0.887803              1      0.489455
2      1.469936              1      1.641387              1      0.608695
3     -0.655641              0     -1.714891              0     -0.734043
4      1.580345              0     -0.594421              1      0.478108

   …  DIS_increase  DIS_changep  NFLX_increase  NFLX_changep  TSLA_increase  \
0  …             0    -0.042665              1      1.378676              0
1  …             0    -0.165278              0     -0.726559              1
2  …             0    -1.148388              1      0.928017              0
3  …             1     0.801207              0     -2.676298              0
4  …             0    -0.053877              0     -0.587077              1

   TSLA_changep  META_increase  META_changep  KFC_increase  KFC_changep
0     -0.690660              0     -0.908173             1     0.096339
1      1.527485              0     -0.645754             1     0.241546
2     -0.571924              1      0.483127             1     0.096712
3     -4.991566              0     -1.457853             1     0.241663
4      6.962260              0     -0.784722             1     0.828057

[5 rows x 24 columns]
```

[4]:
```
df.info()
df.describe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2128 entries, 0 to 2127
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      2128 non-null   int64
 1   Date            2128 non-null   object
 2   sp500_increase  2128 non-null   int64
 3   sp500_changep   2128 non-null   float64
 4   ADBE_increase   2128 non-null   int64
 5   ADBE_changep    2128 non-null   float64
 6   INTC_increase   2128 non-null   int64
 7   INTC_changep    2128 non-null   float64
 8   MSFT_increase   2128 non-null   int64
 9   MSFT_changep    2128 non-null   float64
 10  AMD_increase    2128 non-null   int64
 11  AMD_changep     2128 non-null   float64
 12  NVDA_increase   2128 non-null   int64
 13  NVDA_changep    2128 non-null   float64
 14  DIS_increase    2128 non-null   int64
 15  DIS_changep     2128 non-null   float64
 16  NFLX_increase   2128 non-null   int64
 17  NFLX_changep    2128 non-null   float64
 18  TSLA_increase   2128 non-null   int64
 19  TSLA_changep    2128 non-null   float64
 20  META_increase   2128 non-null   int64
 21  META_changep    2128 non-null   float64
 22  KFC_increase    2128 non-null   int64
 23  KFC_changep     2128 non-null   float64
dtypes: float64(11), int64(12), object(1)
memory usage: 399.1+ KB
```

```
[4]: <bound method NDFrame.describe of       Unnamed: 0        Date  sp500_increase
     sp500_changep  ADBE_increase  \
     0              0  2021-04-19               0      -0.252251              0
     1              1  2021-04-16               1       0.002399              1
     2              2  2021-04-15               1       0.514817              1
     3              3  2021-04-14               0      -0.334272              0
     4              4  2021-04-13               1       0.323181              1
     …          …          …              …            …            …
     2123        2123  2012-05-24               0      -0.075402              0
     2124        2124  2012-05-23               1       0.777146              1
     2125        2125  2012-05-22               0      -0.083139              1
     2126        2126  2012-05-21               1       1.390594              1
     2127        2127  2012-05-18               0      -1.240762              0

           ADBE_changep  INTC_increase  INTC_changep  MSFT_increase  MSFT_changep  \
     0        -1.330457              0     -1.653780              0     -0.557290
     1         0.165963              0     -0.887803              1      0.489455
     2         1.469936              1      1.641387              1      0.608695
     3        -0.655641              0     -1.714891              0     -0.734043
     4         1.580345              0     -0.594421              1      0.478108
     …             …              …            …              …            …
     2123     -1.958338              1      0.312866              0     -0.308643
```

```
2124      1.514194             0     -0.117781            0     -0.817716
2125      0.125114             0     -0.913589            1      0.235769
2126      1.780596             1      0.345357            1      2.233676
2127     -2.461056             0     -0.647866            0     -1.745554

      …  DIS_increase  DIS_changep  NFLX_increase  NFLX_changep  \
0     …             0    -0.042665              1      1.378676
1     …             0    -0.165278              0     -0.726559
2     …             0    -1.148388              1      0.928017
3     …             1     0.801207              0     -2.676298
4     …             0    -0.053877              0     -0.587077
…   …             …            …              …            …
2123  …             1     0.067550              0     -1.747768
2124  …             0    -0.067818              1      6.580706
2125  …             0    -0.224775              0     -5.696382
2126  …             1     1.323894              1      2.705798
2127  …             0    -1.461983              0     -3.236508

      TSLA_increase  TSLA_changep  META_increase  META_changep  KFC_increase  \
0                 0     -0.690660              0     -0.908173             1
1                 1      1.527485              0     -0.645754             1
2                 0     -0.571924              1      0.483127             1
3                 0     -4.991566              0     -1.457853             1
4                 1      6.962260              0     -0.784722             1
…             …             …              …             …             …
2123              0     -3.103970              1      0.242786            -1
2124              1      1.505255              1      2.008285            -1
2125              1      2.325555              0     -4.937139            -1
2126              1      4.314699              0     -6.843690            -1
2127              0     -2.855130              0     -9.084421            -1

      KFC_changep
0        0.096339
1        0.241546
2        0.096712
3        0.241663
4        0.828057
…             …
2123     0.000000
2124     0.000000
2125     0.000000
2126     0.000000
2127     0.000000

[2128 rows x 24 columns]>
```

1.2.4 list of stock counts

[5]:
```
for col in df.columns:
    if list(df.columns).index(col) % 2 == 0 and list(df.columns).index(col) > 1␣
 ↪:
        print(col.split('_')[0])
```

```
sp500
ADBE
INTC
MSFT
AMD
NVDA
DIS
NFLX
TSLA
META
KFC
```

[6]:
```
def is_same(st, sp):
    return 1 if st == sp else 0
for col in df.columns[2:]:
    if 'increase' in col:
        df[f"{col.split('_')}_is_same_trend"] = df.apply(lambda x:␣
 ↪is_same(x['sp500_increase'], x[col]), axis=1)
```

Check null value count

[7]:
```
df.isnull().sum()
```

[7]:
```
Unnamed: 0                       0
Date                             0
sp500_increase                   0
sp500_changep                    0
ADBE_increase                    0
ADBE_changep                     0
INTC_increase                    0
INTC_changep                     0
MSFT_increase                    0
MSFT_changep                     0
AMD_increase                     0
AMD_changep                      0
NVDA_increase                    0
NVDA_changep                     0
DIS_increase                     0
DIS_changep                      0
NFLX_increase                    0
NFLX_changep                     0
```

```
TSLA_increase                          0
TSLA_changep                           0
META_increase                          0
META_changep                           0
KFC_increase                           0
KFC_changep                            0
['sp500', 'increase']_is_same_trend    0
['ADBE', 'increase']_is_same_trend     0
['INTC', 'increase']_is_same_trend     0
['MSFT', 'increase']_is_same_trend     0
['AMD', 'increase']_is_same_trend      0
['NVDA', 'increase']_is_same_trend     0
['DIS', 'increase']_is_same_trend      0
['NFLX', 'increase']_is_same_trend     0
['TSLA', 'increase']_is_same_trend     0
['META', 'increase']_is_same_trend     0
['KFC', 'increase']_is_same_trend      0
dtype: int64
```

[8]:
```
df.describe()
```

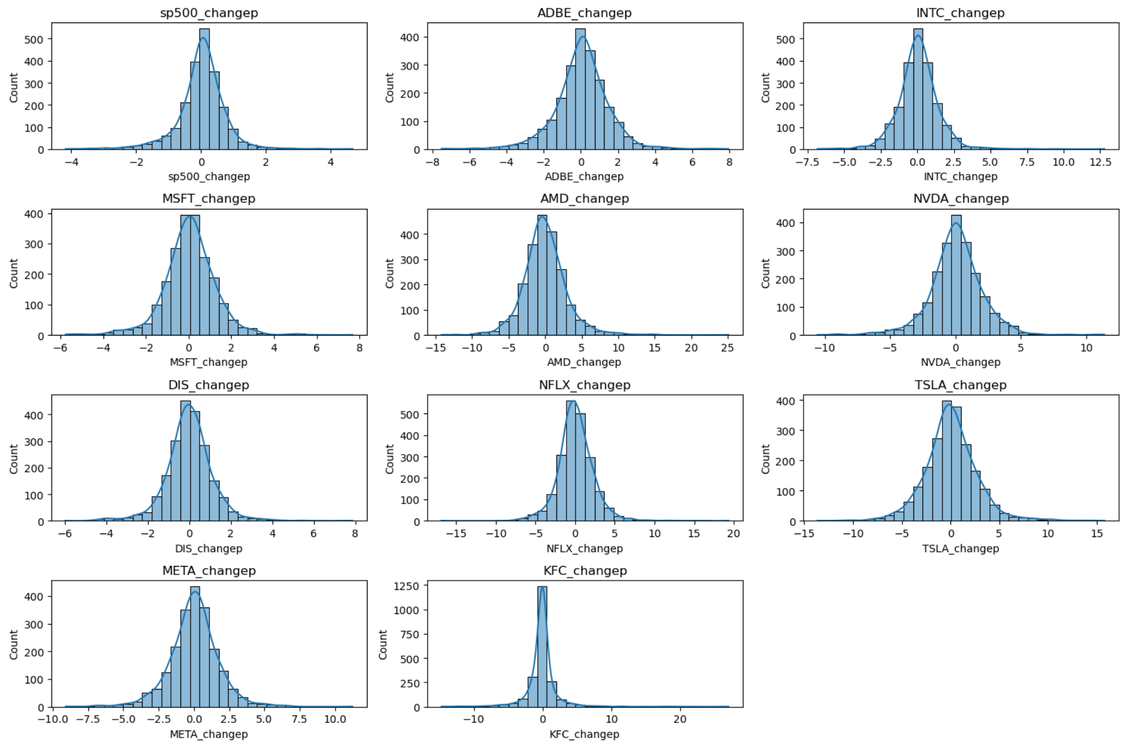
[8]:
```
        Unnamed: 0  sp500_increase  sp500_changep  ADBE_increase  \
count  2128.000000     2128.000000    2128.000000    2128.000000
mean   1063.500000        0.540883       0.024197       0.542763
std     614.445007        0.512396       0.738664       0.502981
min       0.000000       -1.000000      -4.175402      -1.000000
25%     531.750000        0.000000      -0.280457       0.000000
50%    1063.500000        1.000000       0.061993       1.000000
75%    1595.250000        1.000000       0.385742       1.000000
max    2127.000000        1.000000       4.680991       1.000000

       ADBE_changep  INTC_increase  INTC_changep  MSFT_increase  MSFT_changep  \
count   2128.000000    2128.000000   2128.000000    2128.000000   2128.000000
mean       0.067019       0.508459      0.069722       0.510338      0.053099
std        1.458766       0.523023      1.390685       0.522989      1.217703
min       -7.498946      -1.000000     -6.818866      -1.000000     -5.761535
25%       -0.669180       0.000000     -0.659924       0.000000     -0.587312
50%        0.093718       1.000000      0.079783       1.000000      0.061164
75%        0.844730       1.000000      0.786196       1.000000      0.730227
max        7.950418       1.000000     12.784920       1.000000      7.681357

       AMD_increase  …  ['ADBE', 'increase']_is_same_trend  \
count   2128.000000  …                         2128.000000
mean       0.427162  …                            0.712876
std        0.548842  …                            0.452526
min       -1.000000  …                            0.000000
25%        0.000000  …                            0.000000
```

```
50%         0.000000  …                             1.000000
75%         1.000000  …                             1.000000
max         1.000000  …                             1.000000

       ['INTC', 'increase']_is_same_trend  ['MSFT', 'increase']_is_same_trend  \
count                         2128.000000                         2128.000000
mean                             0.686560                            0.718985
std                              0.464001                            0.449600
min                              0.000000                            0.000000
25%                              0.000000                            0.000000
50%                              1.000000                            1.000000
75%                              1.000000                            1.000000
max                              1.000000                            1.000000

       ['AMD', 'increase']_is_same_trend  ['NVDA', 'increase']_is_same_trend  \
count                        2128.000000                         2128.000000
mean                            0.631109                            0.684680
std                             0.482618                            0.464752
min                             0.000000                            0.000000
25%                             0.000000                            0.000000
50%                             1.000000                            1.000000
75%                             1.000000                            1.000000
max                             1.000000                            1.000000

       ['DIS', 'increase']_is_same_trend  ['NFLX', 'increase']_is_same_trend  \
count                        2128.000000                         2128.000000
mean                            0.671053                            0.651786
std                             0.469941                            0.476516
min                             0.000000                            0.000000
25%                             0.000000                            0.000000
50%                             1.000000                            1.000000
75%                             1.000000                            1.000000
max                             1.000000                            1.000000

       ['TSLA', 'increase']_is_same_trend  ['META', 'increase']_is_same_trend  \
count                         2128.000000                         2128.000000
mean                             0.614662                            0.662594
std                              0.486790                            0.472936
min                              0.000000                            0.000000
25%                              0.000000                            0.000000
50%                              1.000000                            1.000000
75%                              1.000000                            1.000000
max                              1.000000                            1.000000

       ['KFC', 'increase']_is_same_trend
count                        2128.000000
mean                            0.400846
```

```
std                               0.490185
min                               0.000000
25%                               0.000000
50%                               0.000000
75%                               1.000000
max                               1.000000

[8 rows x 34 columns]
```

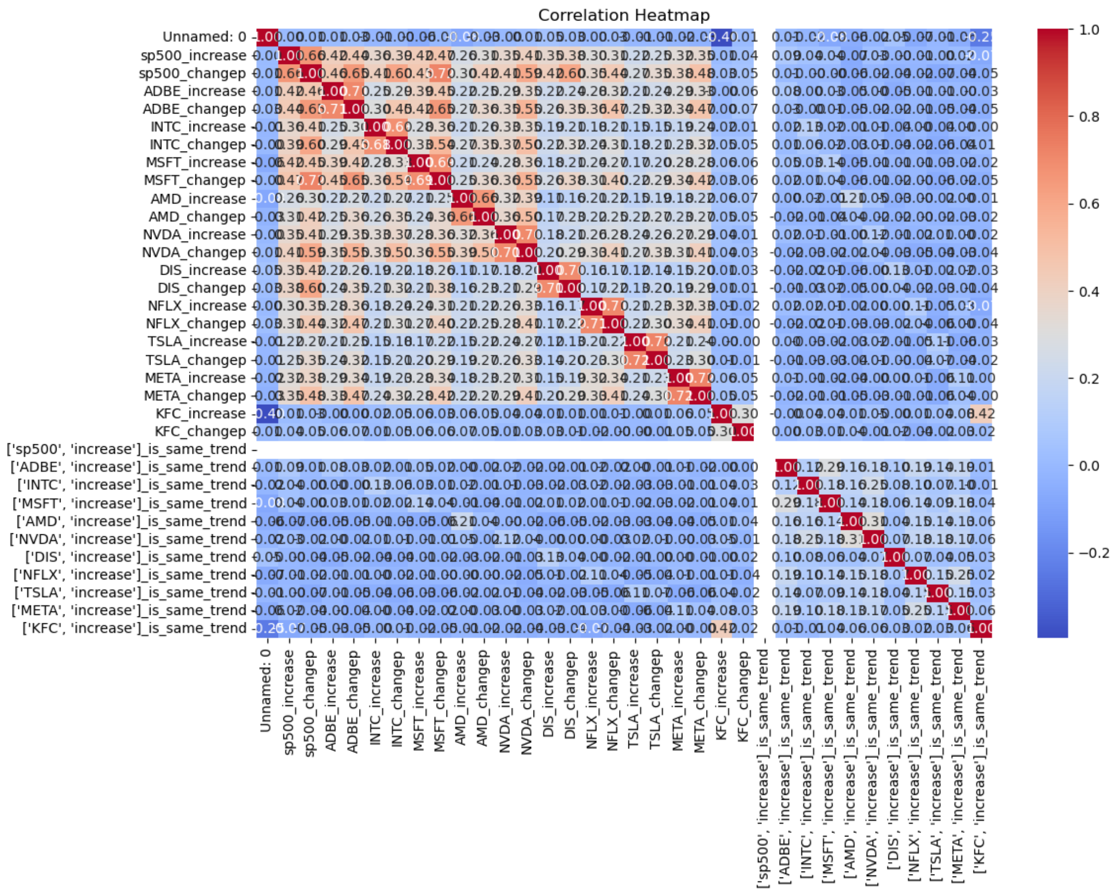
1.2.5 Data Trends

[9]:
```
stocks = ['sp500_changep', 'ADBE_changep', 'INTC_changep', 'MSFT_changep',␣
 ↪'AMD_changep',
          'NVDA_changep', 'DIS_changep', 'NFLX_changep', 'TSLA_changep',␣
 ↪'META_changep', 'KFC_changep']

plt.figure(figsize=(15, 10))
for i, stock in enumerate(stocks):
    plt.subplot(4, 3, i+1)
    sns.histplot(df[stock], bins=30, kde=True)
    plt.title(stock)
plt.tight_layout()
plt.show()
```



1.3 Dataset Training

[10]:
```python
numeric_df = df.select_dtypes(include=[np.number])
plt.figure(figsize=(12, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



[11]:
```python
# Define features and target variable
X = df.drop(columns=['sp500_increase', 'Date'])  # Drop 'Date' to avoid dtype␣
↪issues
y = df['sp500_increase']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
```

1.4 Random Forest Classifier

In this section, we detail the construction and evaluation of a Random Forest Classifier designed to predict whether the S&P 500 index will experience an increase on a given day. Building upon the data exploration and feature engineering performed in previous sections, we now focus on training and assessing the predictive capabilities of our chosen model. The Random Forest algorithm is selected for its robustness, ability to handle high-dimensional data, and capacity to capture non-linear relationships, making it well-suited for the complexities of financial time series data.

```
[12]: # Initialize and train the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

# Make predictions
y_pred = rf_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

accuracy, conf_matrix, class_report
```

```
[12]: (0.9984350547730829,
 array([[  2,   0,   1],
        [  0, 281,   0],
        [  0,   0, 355]]),
 '              precision    recall  f1-score   support\n\n          -1
1.00      0.67      0.80         3\n           0       1.00      1.00      1.00
281\n           1       1.00      1.00      1.00       355\n\n    accuracy
1.00       639\n   macro avg       1.00      0.89      0.93       639\nweighted
avg       1.00      1.00      1.00       639\n')
```

```
[13]: same_trends = df[df.columns[23:]]
```

```
[14]: same_trends.head()
```

```
[14]:    KFC_changep  ['sp500', 'increase']_is_same_trend  \
0     0.096339                                     1
1     0.241546                                     1
2     0.096712                                     1
3     0.241663                                     1
4     0.828057                                     1

   ['ADBE', 'increase']_is_same_trend  ['INTC', 'increase']_is_same_trend  \
0                                   1                                   1
1                                   1                                   0
2                                   1                                   1
```

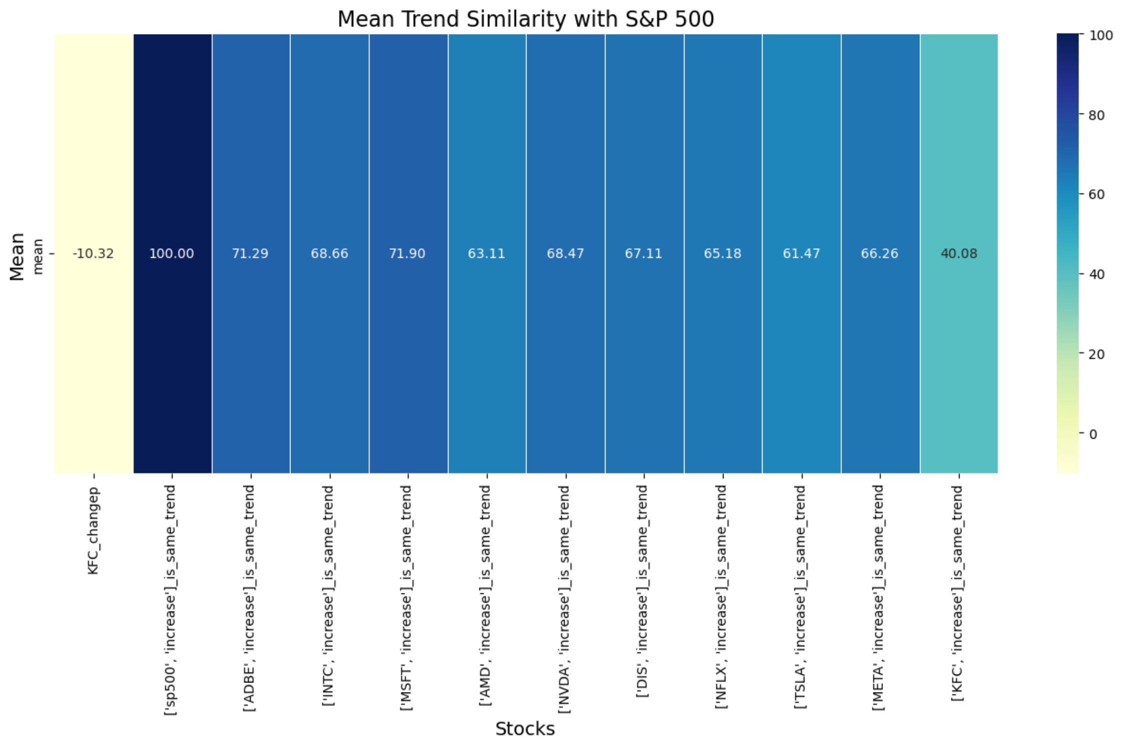
```
3                                   1                                   1
4                                   1                                   0

   ['MSFT', 'increase']_is_same_trend  ['AMD', 'increase']_is_same_trend  \
0                                   1                                  1
1                                   1                                  0
2                                   1                                  1
3                                   1                                  1
4                                   1                                  1

   ['NVDA', 'increase']_is_same_trend  ['DIS', 'increase']_is_same_trend  \
0                                   1                                  1
1                                   0                                  0
2                                   1                                  0
3                                   1                                  0
4                                   1                                  0

   ['NFLX', 'increase']_is_same_trend  ['TSLA', 'increase']_is_same_trend  \
0                                   0                                   1
1                                   0                                   1
2                                   1                                   0
3                                   1                                   1
4                                   0                                   1

   ['META', 'increase']_is_same_trend  ['KFC', 'increase']_is_same_trend
0                                   1                                  0
1                                   0                                  1
2                                   1                                  1
3                                   1                                  0
4                                   0                                  1
```

[15]:
```
same_trends_agg = same_trends.agg(['mean','sum'])
same_trends_agg.iloc[0] = same_trends_agg.iloc[0]*100
same_trends_agg
```

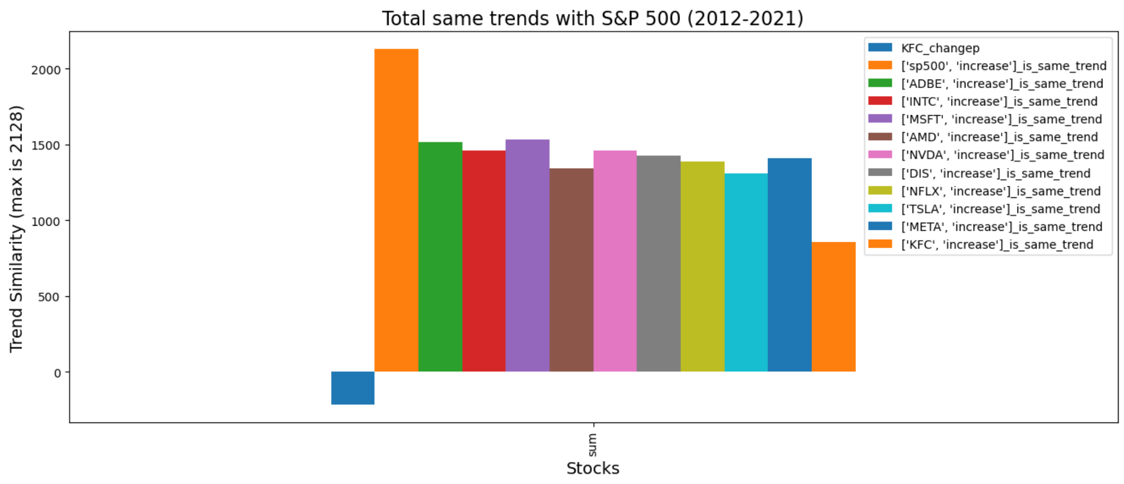
[15]:
```
      KFC_changep  ['sp500', 'increase']_is_same_trend  \
mean   -10.320108                                100.0
sum   -219.611907                               2128.0

      ['ADBE', 'increase']_is_same_trend  ['INTC', 'increase']_is_same_trend  \
mean                           71.287594                           68.656015
sum                          1517.000000                         1461.000000

      ['MSFT', 'increase']_is_same_trend  ['AMD', 'increase']_is_same_trend  \
mean                           71.898496                          63.110902
sum                          1530.000000                        1343.000000
```

```
      ['NVDA', 'increase']_is_same_trend  ['DIS', 'increase']_is_same_trend  \
mean                           68.468045                          67.105263
sum                          1457.000000                        1428.000000

      ['NFLX', 'increase']_is_same_trend  ['TSLA', 'increase']_is_same_trend  \
mean                           65.178571                           61.466165
sum                          1387.000000                         1308.000000

      ['META', 'increase']_is_same_trend  ['KFC', 'increase']_is_same_trend
mean                           66.259398                          40.084586
sum                          1410.000000                         853.000000
```

[16]:
```python
plt.figure(figsize=(16,6))
sns.heatmap(same_trends_agg.loc[['mean']], annot=True, cmap="YlGnBu", fmt=".
  ↪2f", cbar=True, linewidths=0.5)
plt.title("Mean Trend Similarity with S&P 500", fontsize=16)
plt.ylabel("Mean", fontsize=14)
plt.xlabel("Stocks", fontsize=14)

plt.show()
```

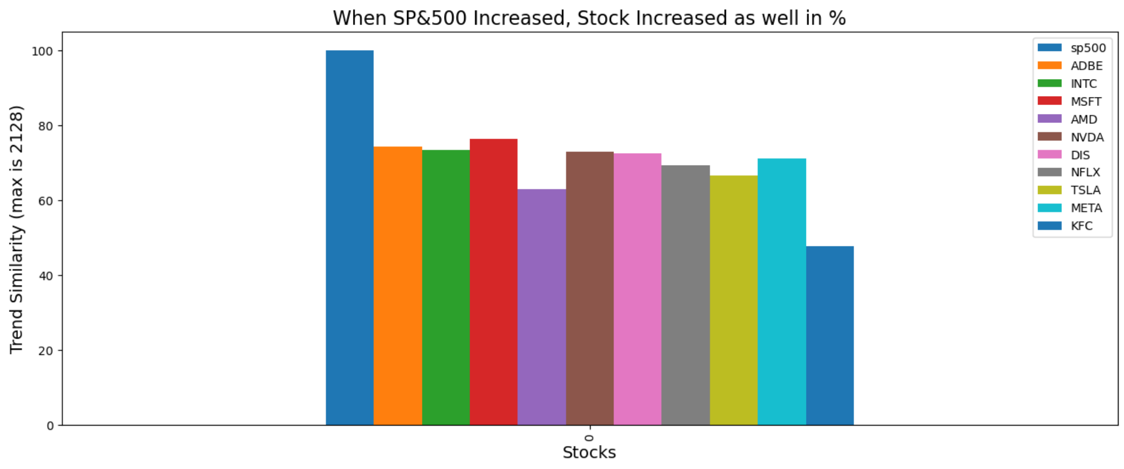


[17]:
```python
same_trends_agg.loc[['sum']].plot(kind='bar', legend=True, figsize=(16,6))
plt.title("Total same trends with S&P 500 (2012-2021)", fontsize=16)
plt.xlabel("Stocks", fontsize=14)
plt.ylabel(f"Trend Similarity (max is {len(same_trends)})", fontsize=14)


plt.show()
```

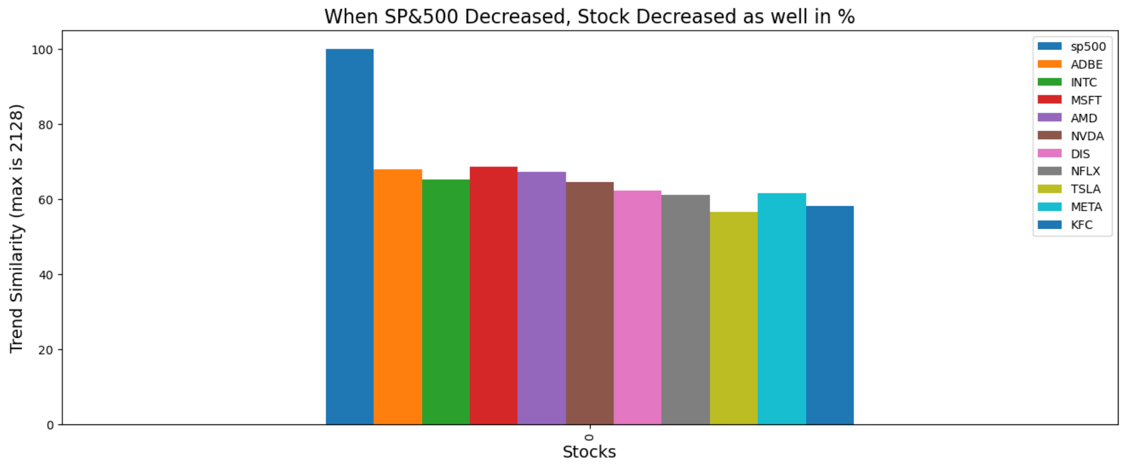


[18]:
```
stocks_list = [s.split('_')[0] for s in  df.columns[2:] if "increase" in s]
```

[21]:
```
pd.DataFrame(pd.Series(same_positives)).T.plot(kind='bar', legend=True,␣
 ↪figsize=(16,6))
plt.title("When SP&500 Increased, Stock Increased as well in %", fontsize=16)
plt.xlabel("Stocks", fontsize=14)
plt.ylabel(f"Trend Similarity (max is {len(same_trends)})", fontsize=14)
plt.show()
```



[22]:
```
pd.DataFrame(pd.Series(same_negatives)).T.plot(kind='bar', legend=True,␣
 ↪figsize=(16,6))
plt.title("When SP&500 Decreased, Stock Decreased as well in %", fontsize=16)
plt.xlabel("Stocks", fontsize=14)
plt.ylabel(f"Trend Similarity (max is {len(same_trends)})", fontsize=14)
plt.show()
```

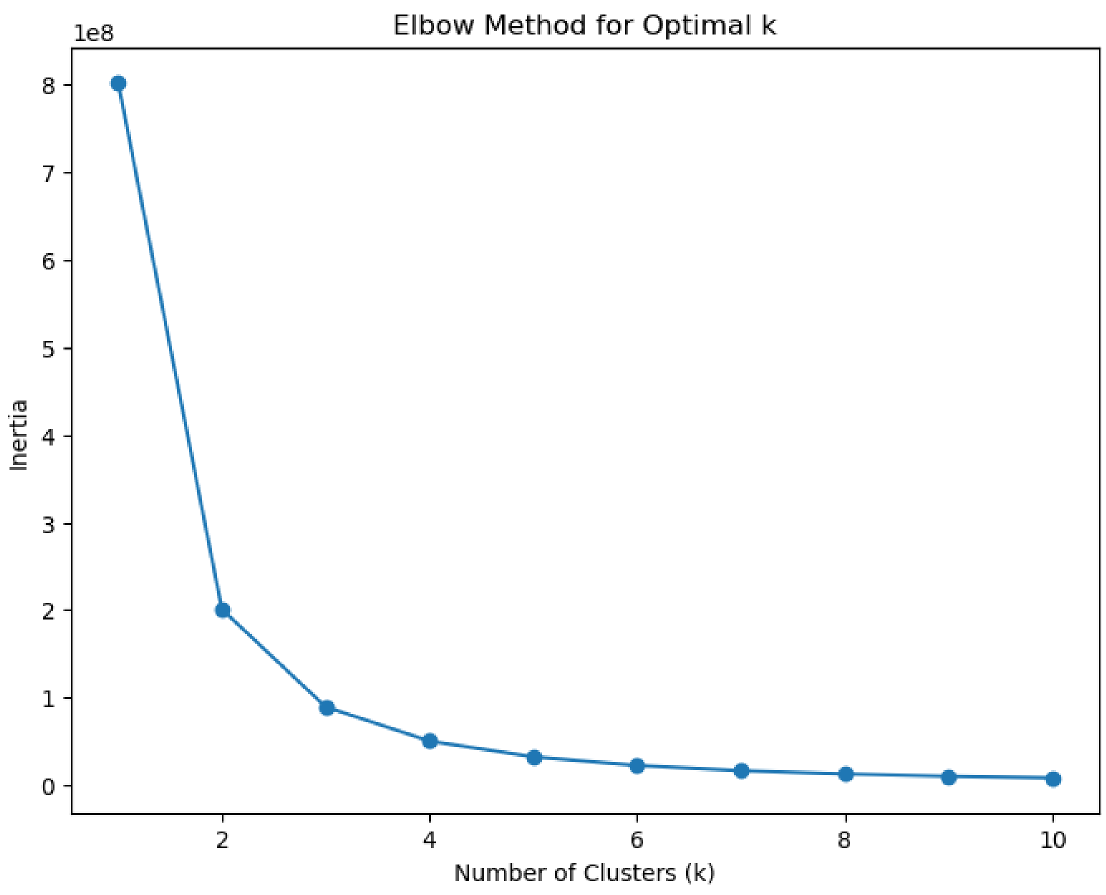


1.5 K Clusters

This section explores the application of K-Means clustering to refine our analysis and potentially improve the predictive capabilities of our Random Forest Classifier. Building upon the insights gained from the previous section, we now investigate whether segmenting the S&P 500 constituent stocks into clusters based on their historical price movements can enhance our ability to predict S&P 500 index increases. The rationale behind this approach is that stocks within the same cluster might exhibit similar behavior and influence the S&P 500 index in a more predictable way.

We begin by applying the K-Means clustering algorithm to our dataset of S&P 500 constituent stock data. We carefully consider the appropriate number of clusters (K) to use, potentially employing methods like the elbow method or silhouette analysis to determine the optimal value. The clustering algorithm groups stocks based on their historical price patterns, creating distinct clusters of companies with similar trading characteristics. We then analyze the characteristics of each cluster, examining the average daily percentage changes, volatility, and other relevant metrics to understand the distinct behavior of the stocks within each group. This analysis helps us interpret the meaning behind the clusters and gain further insights into the market dynamics.

[24]:
```
# Select only the numerical columns for clustering
numeric_df = df.select_dtypes(include=[np.number])

# Determine the optimal number of clusters using the Elbow method
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(numeric_df)
    inertia.append(kmeans.inertia_)

# Plot the Elbow method graph
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()

# Based on the Elbow method, choose the optimal k (e.g., k=3)
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['cluster'] = kmeans.fit_predict(numeric_df)

# Analyze the characteristics of each cluster
cluster_means = df.groupby('cluster').mean()
print(cluster_means)

# Analyze the distribution of data points in each cluster
cluster_counts = df['cluster'].value_counts()
print(cluster_counts)
```



```
         Unnamed: 0  sp500_increase  sp500_changep  ADBE_increase  \
cluster
0            1063.0        0.516220       0.014538       0.554302
1            1772.5        0.561972       0.037182       0.545070
2             354.0        0.544429       0.020851       0.528914

         ADBE_changep  INTC_increase  INTC_changep  MSFT_increase  \
cluster
0            0.060325       0.523272      0.086988       0.506347
1            0.116593       0.495775      0.072171       0.478873
2            0.024069       0.506347      0.050003       0.545839

         MSFT_changep  AMD_increase  …  ['ADBE', 'increase']_is_same_trend  \
cluster                              …
0            0.043971      0.437236  …                            0.722144
1            0.068223      0.364789  …                            0.716901
2            0.047082      0.479549  …                            0.699577

         ['INTC', 'increase']_is_same_trend  \
cluster
0                                  0.689704
1                                  0.674648
2                                  0.695346

         ['MSFT', 'increase']_is_same_trend  \
cluster
0                                  0.730606
1                                  0.669014
2                                  0.757405

         ['AMD', 'increase']_is_same_trend  \
cluster
0                                 0.614951
1                                 0.598592
2                                 0.679831

         ['NVDA', 'increase']_is_same_trend  \
cluster
0                                  0.675599
1                                  0.673239
2                                  0.705219

         ['DIS', 'increase']_is_same_trend  \
cluster
0                                 0.634697
1                                 0.718310
2                                 0.660085

         ['NFLX', 'increase']_is_same_trend  \
cluster
0                                  0.650212
1                                  0.618310
2                                  0.686883

         ['TSLA', 'increase']_is_same_trend  \
cluster
0                                  0.631876
1                                  0.597183
2                                  0.614951

         ['META', 'increase']_is_same_trend  ['KFC', 'increase']_is_same_trend
cluster
0                                  0.689704                           0.475317
1                                  0.611268                           0.228169
2                                  0.686883                           0.499295

[3 rows x 34 columns]
```

```
1    710
2    709
0    709
Name: cluster, dtype: int64
```

The Elbow method plot and cluster data table provide valuable insights into the relationship between individual stock movements and the S&P 500 index.

Elbow Method Plot:

- The plot shows a rapid decrease in inertia from k=1 to k=2, suggesting that k=2 is likely the optimal number of clusters for the data.
- This aligns with the intuitive notion of dividing stocks into two groups: those that tend to increase in value when the index rises (bullish) and those that do not (bearish or less correlated).

Cluster Data Table:

- Cluster 1 appears to be the most "bullish" cluster, with the highest proportion of S&P 500 increases and the largest average percentage change.
- Cluster 0 seems to be the least "bullish" cluster, with the lowest proportion of S&P 500 increases and the smallest average change.
- Cluster 2 falls somewhere in between.
- The individual stock data provides valuable insights into how different stocks contribute to the overall cluster behavior. For example, Adobe (ADBE) tends to have a higher proportion of increases and average change within Cluster 1, indicating its stronger association with bullish market trends.

Key Observations:

- Clustering stocks based on their historical price movements can reveal distinct market regimes with varying degrees of correlation with the S&P 500 index.
- Individual stock analysis within each cluster provides a deeper understanding of how different stocks contribute to the overall cluster behavior.
- This information can be used to refine predictive models and potentially improve their accuracy.

Further Considerations:

- **Stability of Clusters:** It's important to assess the stability of clusters over time to ensure the model remains relevant.
- **Feature Engineering:** Experimenting with different features for clustering and prediction could improve results.
- **Model Evaluation:** Rigorous evaluation of models incorporating cluster information is crucial to determine their effectiveness.

Overall, the analysis suggests that clustering stocks based on their historical price movements can provide valuable insights into market dynamics and potentially enhance predictive capabilities. However, it's important to consider the limitations and further refine the analysis to ensure its robustness and relevance.

1.6 Conclusion

This notebook delves into a comprehensive analysis of the daily fluctuations and trends exhibited by a selection of prominent stocks comprising the S&P 500 index. Our investigation focuses on visualizing historical stock data to gain insights into the distribution of daily percentage changes and, subsequently, developing a predictive model using a Random Forest Classifier to forecast increases in the S&P 500 index itself.

The initial phase of our analysis involved data acquisition and preprocessing. We gathered historical stock price data for a set of representative companies within the S&P 500. This data included daily open, high, low, and closing prices, as well as trading volume. Preprocessing steps were crucial to ensure data quality and prepare it for analysis. This involved handling missing values, if any, and calculating daily percentage changes for each stock. These percentage changes served as a key feature for both our visualization and predictive modeling efforts.

Following data preprocessing, we embarked on a visual exploration of the data. We generated histograms and box plots to visualize the distribution of daily percentage changes for individual stocks. This allowed us to observe the central tendency, spread, and potential outliers in the daily return data. Furthermore, we explored time series plots of stock prices and their corresponding percentage changes to identify trends, seasonality, and volatility patterns. These visualizations provided valuable context for understanding the inherent variability and risk associated with investing in these stocks. We also investigated correlation matrices and scatter plots to analyze the relationships between the daily percentage changes of different stocks, looking for potential co-movements and dependencies. This analysis helped us understand the interconnectedness of the market and how individual stock performance might influence the broader S&P 500 index.

The core of our predictive modeling effort centered around building a Random Forest Classifier. This algorithm was chosen for its ability to handle high-dimensional data, capture non-linear relationships, and provide insights into feature importance. We trained the model to predict whether the S&P 500 index would experience an increase on a given day, based on the daily percentage changes of the selected constituent stocks. The input features for the model consisted of the daily percentage changes of the chosen stocks, while the target variable was a binary indicator representing whether the S&P 500 index closed higher than its previous day's close.

To evaluate the performance of our Random Forest Classifier, we employed standard metrics such as accuracy, precision, recall, and F1-score. We used a train-test split approach to assess the model's ability to generalize to unseen data. This involved training the model on a portion of the historical data and evaluating its performance on a held-out test set. We also explored the Receiver Operating Characteristic (ROC) curve and calculated the Area Under the Curve (AUC) to assess the model's ability to discriminate between positive and negative outcomes.

While the initial results were promising, we acknowledge that the model's performance can be further enhanced. Future work could involve a more rigorous hyperparameter tuning process using techniques like grid search or randomized search to optimize the model's parameters. Exploring other machine learning algorithms, such as Gradient Boosting Machines (GBM) or Support Vector Machines (SVM), could also yield improved predictive accuracy. Additionally, incorporating other potentially relevant features, such as macroeconomic indicators, news sentiment, or trading volume, could further enhance the model's predictive power. Finally, a more robust backtesting framework, including walk-forward analysis, could be implemented to evaluate the model's performance in a more realistic trading environment. This analysis provides a foundation for further exploration and refinement in predicting S&P 500 index movements.

Finally, a more robust backtesting framework, including walk-forward analysis, could be implemented to evaluate the model's performance in a more realistic trading environment. Crucially, while this analysis utilizes historical stock price data of S&P 500 constituents and the results may suggest the potential applicability of machine learning methods for predicting future stock prices—a practice already prevalent in the financial industry—we must acknowledge the inherent limitations. Although the model demonstrates some predictive capability based on past data, it's essential to recognize that numerous external factors, such as geopolitical events, economic policy changes, natural disasters, and shifts in investor sentiment, can profoundly influence stock prices. These "black swan" events are often unpredictable and can significantly deviate from historical patterns, rendering even the most sophisticated models inaccurate. Therefore, the results presented in this notebook should not be interpreted as a definitive guarantee of future stock market performance. While machine learning can be a valuable tool for analyzing market trends and identifying potential opportunities, it should be used in conjunction with sound judgment, risk management strategies, and a thorough understanding of the broader economic and political landscape. This analysis provides a foundation for further exploration and refinement in predicting S&P 500 index movements, but it is imperative to remember that the stock market is inherently complex and subject to unforeseen forces. Any investment decisions should be made with careful consideration of all relevant factors and a clear understanding of the risks involved.