Comprehensive Report on Stock Market Classification

PAN AFRICAN UNIVERSITY

INSTITUTE FOR BASIC SCIENCES, TECHNOLOGY AND INNOVATION

Name: Haruna Jallow

Subject: Data Mining and Knowledge

Major: MSc Data Science

October 30 2023

Contents

1 Abstract

This study aims to create a classification model to predict the "Day Price" variable using historical stock exchange data from the NSE (National Stock Exchange). The research spans four months and employs three different classification models: Naive Bayes, K-Nearest Neighbors (KNN), and Random Forest. Evaluation of these models is based on key performance metrics, including accuracy, precision, recall, and the F1-score. The objective is to develop a reliable predictive model that can assist investors and traders in making informed decisions regarding stock prices.

2 Introduction

This study aims to predict the "Day Price" variable of the National Stock Exchange (NSE) using historical data from four months. We compare three classification models: Naive Bayes, K-Nearest Neighbors (KNN), and Random Forest. These models can help investors and traders anticipate the direction of stock prices based on historical patterns and trends. We evaluate the models' performance using accuracy, precision, recall, and the F1-score. The goal is to provide a reliable tool for data-driven investment decisions in the stock market.

3 Setup and Initialization

3.1 Importing Libraries

I started by importing the necessary Python libraries for data analysis and machine learning.

3.2 Importing Dataset

I loaded the dataset containing historical stock data from the NSE. And I drop the columns that I don't want to include and assign the result to a new variable called selectedData. This is because the variables that i excluded don't have more of empty values.

4 Creating a Binary Label

We created a binary label, where 1 indicates that 'Day Price' is going high, and 0 indicates that 'Day Price' is going low compared to the previous day's price.

Creating A Binary Label :

Where 1 indicates that 'Day Price' is going high, and 0 indicates that 'Day Price' is going low compared to the previous day's price.

```
In [6]: # Define a binary label: 1 for 'Day Price' going high, 0 for 'Day Price' going low
        selected_data['Price Increase'] = np.where(selected_data['Day Price'] > selected_data['Previous'], 1, 0)

In [7]: selected_data.head()
```

Out[7]:

	Date	Name	12m Low	12m High	Day Low	Day High	Day Price	Previous	Price Increase
0	3-Jan-22	Eaagads Ltd	10	15	13.5	13.8	13.5	13.5	0
1	3-Jan-22	Kakuzi Plc	355	427	385	385	385	385	0
2	3-Jan-22	Kapchorua Tea Kenya Plc	80	101	99.5	99.5	99.5	95.5	1
3	3-Jan-22	Limuru Tea Plc	260	360	320	320	320	320	0
4	3-Jan-22	Sasini Plc	16.75	22.6	18.7	18.7	18.7	18.7	0

5 Transformation

I did data transformation by removing; strings, dash symbols, quotation marks etc.

Transformation :

In [8]:
```
# Define a list of column names that you want to include
columns_to_include = ["12m Low", "12m High", "Day Low", "Day High", "Day Price", "Previous"]

# Loop over the column names and apply the same operations to each column
for column in columns_to_include:
  # Replace commas with empty strings
  selected_data[column] = selected_data[column].str.replace(",", "")
  # Replace dashes with np.nan
  selected_data[column] = selected_data[column].replace("-", np.nan)
  # Convert the values to numeric and handle errors
  selected_data[column] = pd.to_numeric(selected_data[column], errors="coerce")
```

In [9]:
```
selected_data.head()
```

Out[9]:

	Date	Name	12m Low	12m High	Day Low	Day High	Day Price	Previous	Price Increase
0	3-Jan-22	Eaagads Ltd	10.00	15.0	13.5	13.8	13.5	13.5	0
1	3-Jan-22	Kakuzi Plc	355.00	427.0	385.0	385.0	385.0	385.0	0
2	3-Jan-22	Kapchorua Tea Kenya Plc	80.00	101.0	99.5	99.5	99.5	95.5	1
3	3-Jan-22	Limuru Tea Plc	260.00	360.0	320.0	320.0	320.0	320.0	0
4	3-Jan-22	Sasini Plc	16.75	22.6	18.7	18.7	18.7	18.7	0

6 Handling Missing Values

Clean missing values using Random Value Imputation Because This the best way to To maintain distrbuation For each feature.

In [10]:
```
# Impute the missing values in Volume, Change with the median of each variable
def impute_median(series):
    return series.fillna(series.median())
selected_data[['Volume', 'Change']] = selected_data[['Volume', 'Change']].apply(impute_median)
```

In [11]:
```
selected_data.sample(5)
```

Out[11]:

	Date	Name	12m Low	12m High	Day Low	Day High	Day Price	Previous	Change	Volume	Price Increase
3275	11-Mar-22	Total Kenya Ltd	22.05	27.50	23.20	23.50	23.25	23.40	-0.15	12100.0	0
3386	15-Mar-22	Eveready East Africa Ltd	0.77	1.27	0.86	0.86	0.86	0.89	-0.03	12900.0	0
1175	26-Jan-22	Trans-Century Plc	1.00	1.66	1.34	1.34	1.34	1.33	0.01	2100.0	1
4173	31-Mar-22	National Bank of Kenya Ltd	4.12	4.12	4.12	4.12	4.12	4.12	-0.01	10800.0	0
6410	25-May-22	BK Group Plc	24.30	40.00	31.05	31.05	31.05	30.00	1.05	100.0	1

7 Exploratory Data Analysis (EDA)

We conducted exploratory data analysis to gain insights into the dataset and visualize the relationships between different features. This analysis included the following visualizations:

- **Basic Descriptive Statistics of the dataset:**
  Getting the basic statistics of the dataset.

Getting Description of the data set :

```
In [12]: selected_data.describe()
```

Out[12]:

	12m Low	12m High	Day Low	Day High	Day Price	Previous	Change	Volume	Price Increase
count	6732.000000	6732.000000	6732.000000	6732.000000	6732.000000	6732.000000	6732.000000	6.732000e+03	6732.000000
mean	86.688598	105.925501	96.248228	96.686641	96.440997	96.413639	0.022941	1.912565e+05	0.259507
std	286.829071	324.290414	309.676291	309.764709	309.715686	309.482751	3.074186	1.097252e+06	0.438397
min	0.170000	0.270000	0.170000	0.180000	0.170000	0.170000	-40.000000	1.000000e+02	0.000000
25%	3.400000	4.600000	3.830000	3.860000	3.830000	3.830000	-0.030000	3.800000e+03	0.000000
50%	9.210000	13.650000	10.975000	11.000000	11.000000	11.000000	-0.010000	1.080000e+04	0.000000
75%	33.000000	45.250000	37.000000	37.762500	37.462500	37.462500	0.010000	2.682500e+04	1.000000
max	1780.000000	2135.000000	2135.000000	2135.000000	2135.000000	2135.000000	195.000000	2.214110e+07	1.000000

- **Getting Information about the data set**
  Getting information about the dataset; date types null values etc.

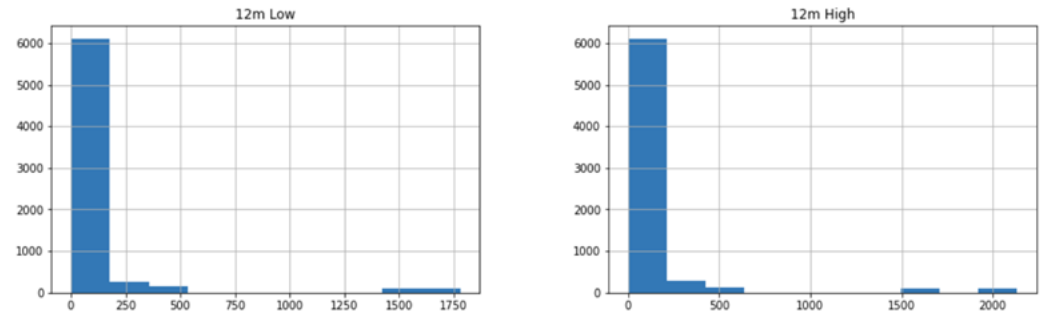
Getting Information about the data set :

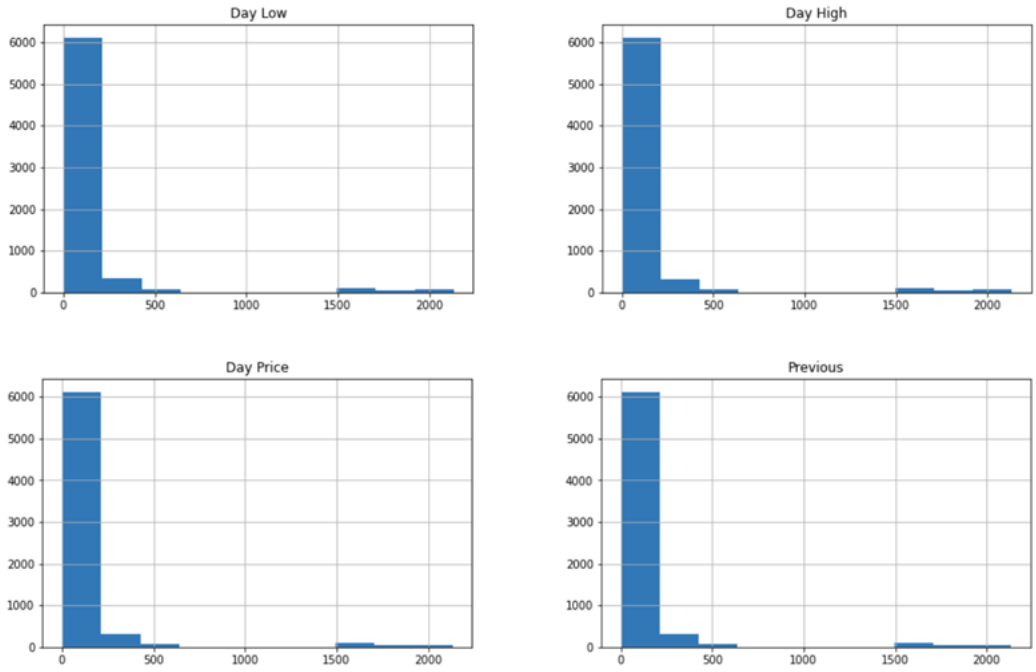
```
In [15]: selected_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6732 entries, 0 to 6731
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            6732 non-null   object
 1   Name            6732 non-null   object
 2   12m Low         6732 non-null   float64
 3   12m High        6732 non-null   float64
 4   Day Low         6732 non-null   float64
 5   Day High        6732 non-null   float64
 6   Day Price       6732 non-null   float64
 7   Previous        6732 non-null   float64
 8   Change          6732 non-null   float64
 9   Volume          6732 non-null   float64
 10  Price Increase  6732 non-null   int32
dtypes: float64(8), int32(1), object(2)
memory usage: 552.4+ KB
```

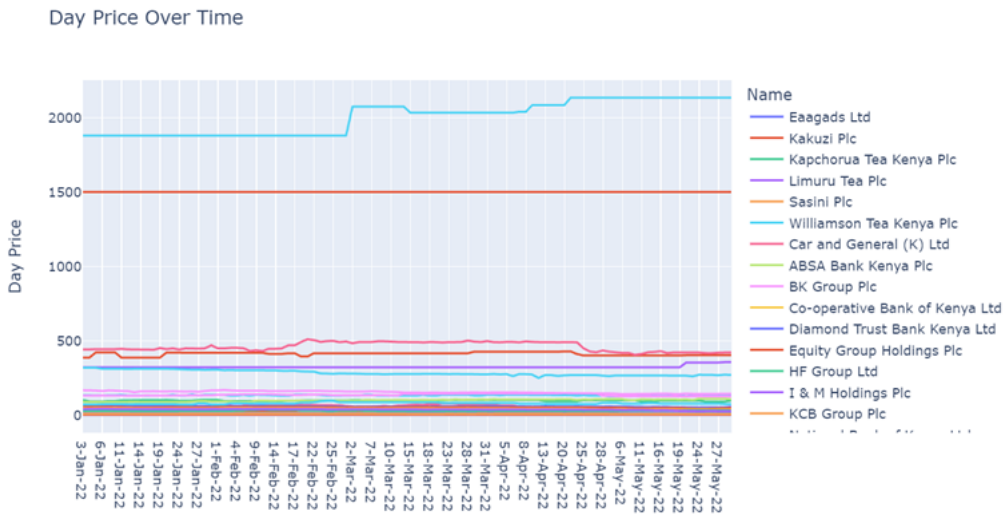
- **Distribution of the Dataset**
  Checking distribution of the data set to identify outlier. As we can see, the variables outliers and their distribution right skewed.

```
In [17]: selected_data.hist(column=["12m Low", "12m High", "Day Low", "Day High", "Day Price", "Previous"], figsize=(16, 16))
plt.show()
```





- **Plots to identify Trends and Correlations**



The line plot reveals some interesting patterns in the data. First, we can see that there is a seasonal variation in the prices of the products, as they tend to increase in the summer months and decrease in the winter months. This could be related to the demand or supply of the products in different seasons.

Second, we can see that there is a difference in the price levels and trends among the companies. For example, Eaagad Ltd has the highest average price and shows a steady increase over time, while Uchumi Supermarket of has the lowest average price and shows a slight decrease over time. This could be related to the quality or popularity of the products/services in the market.

Third, we can see that there are some outliers and fluctuations in the prices of some

products/services. For example, some companies have seen a sharp drop in price in July or so, while some companies have seen a sudden spike in price in November. These outliers could be due to various factors, such as promotions, discounts, errors, or changes in demand or supply

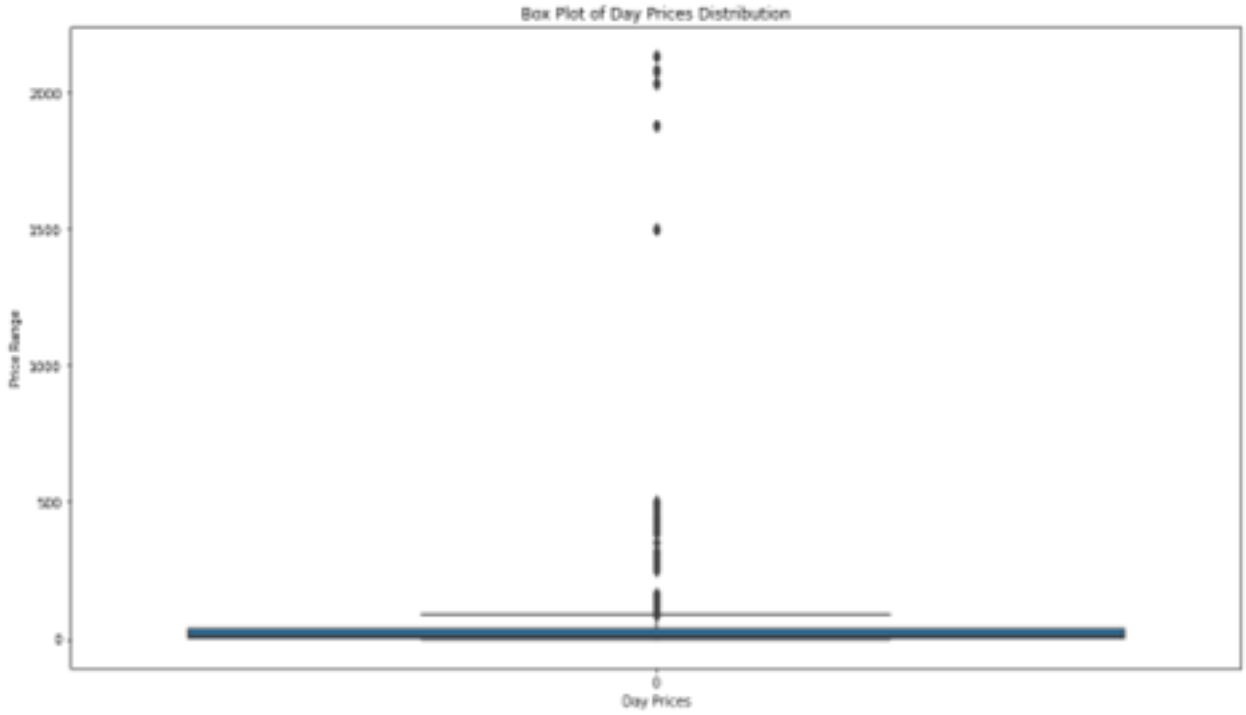
- **Comparison with Previous Day Price**

The scatter plot reveals some interesting patterns in the data. First, we can see that there is a strong positive linear association between the day price and the previous price, which means that products/services with higher previous prices tend to have higher day prices as well. This makes sense, as we would expect products/services with higher demand or quality to maintain their high prices over time.

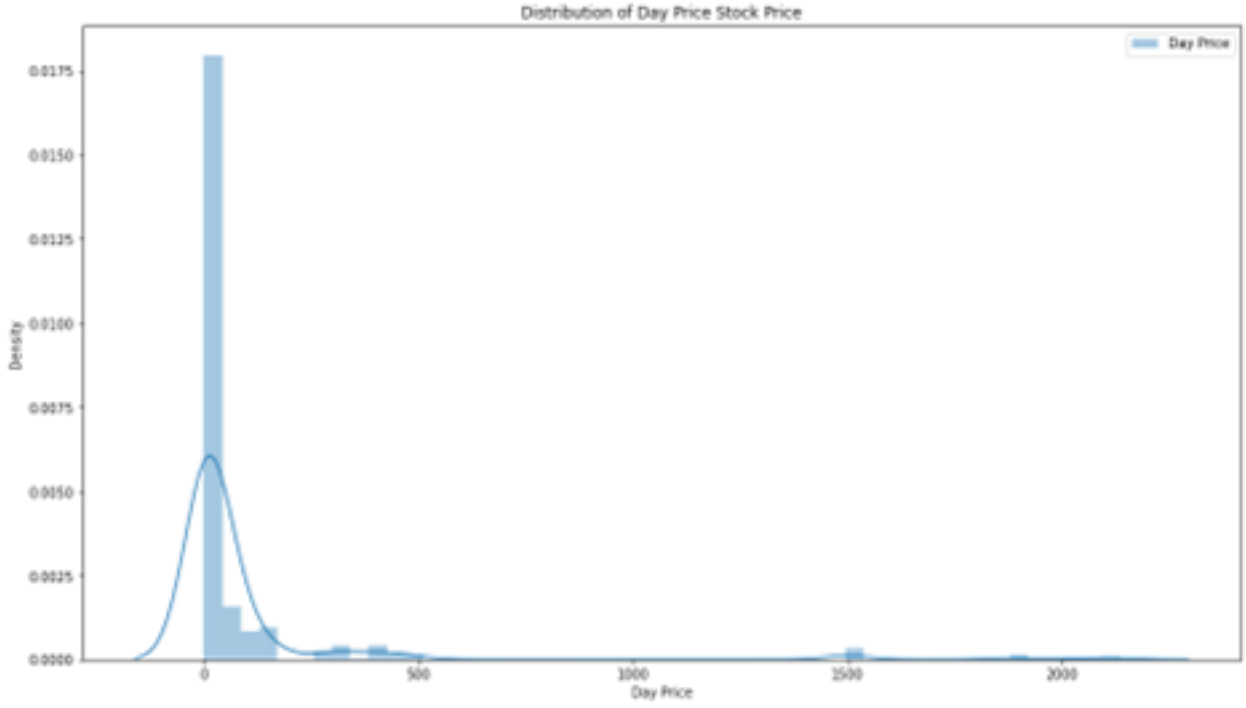
Second, we can see that most of the data points are colored green, which means that most of the products/services have increased their prices from the previous day. This suggests that there is a general trend of inflation or rising demand in the market. However, some outliers are colored red, which means that they have decreased their prices from the previous day. These outliers could be due to various factors, such as promotions, discounts, errors, or changes in supply or demand.

Third, we can see that the color intensity varies across the data points, which means that the magnitude of the price increase also varies. Some products/services have a slight price increase, while others have a significant price increase. This could be related to the different types or categories of products/services, or their different levels of competition or popularity in the market.

- **Distribution of Target Variable**
  Plotting box plot to find the distribution of day price and previous in the data. The box plot shows that there 'Day Price' variables has outliers as seen already

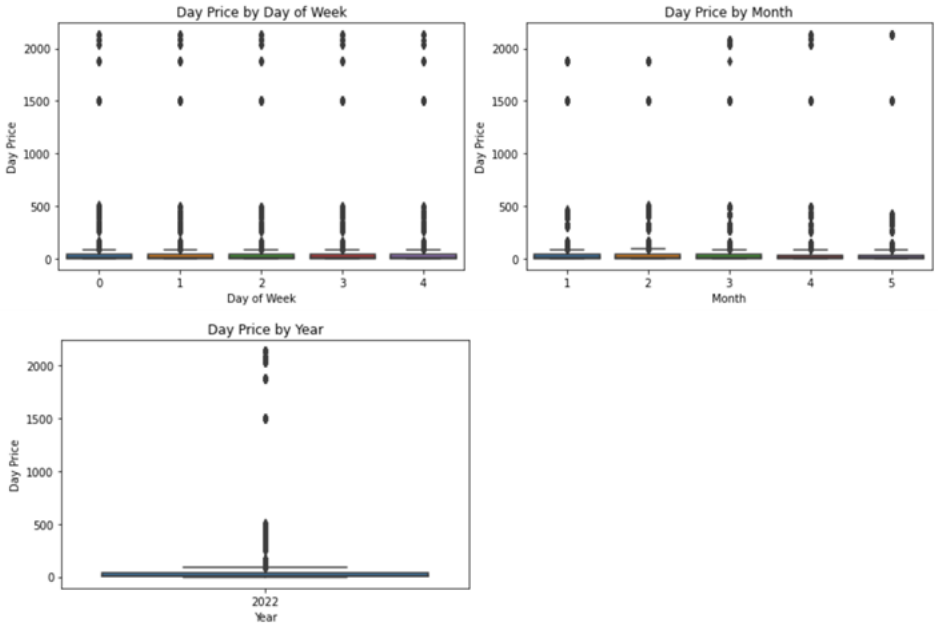


- **Distribution of Target Variable**
  Plotting Distribution plot to find the distribution of day price in the dataset.

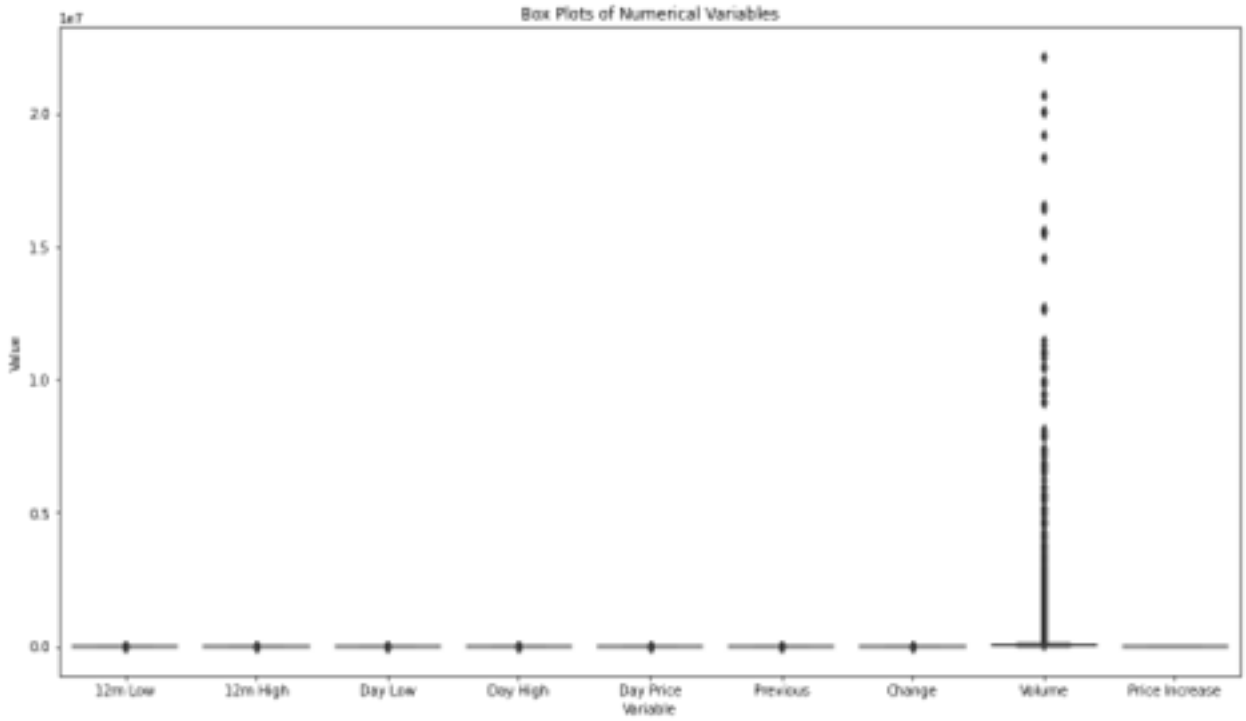


- **Plotting Correlation of the Variables**
  Plotting Correlation plot to find the correlation of attributes. There is high correlation among the variables. This is visiable in the grap below

- **Distribution of Stock Prices by Day of the Week, Month, and Year**



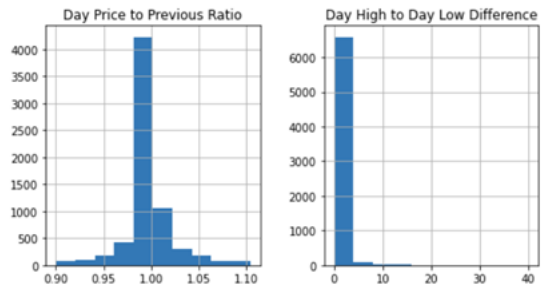
- **Box Plots of the Numerical Variables**



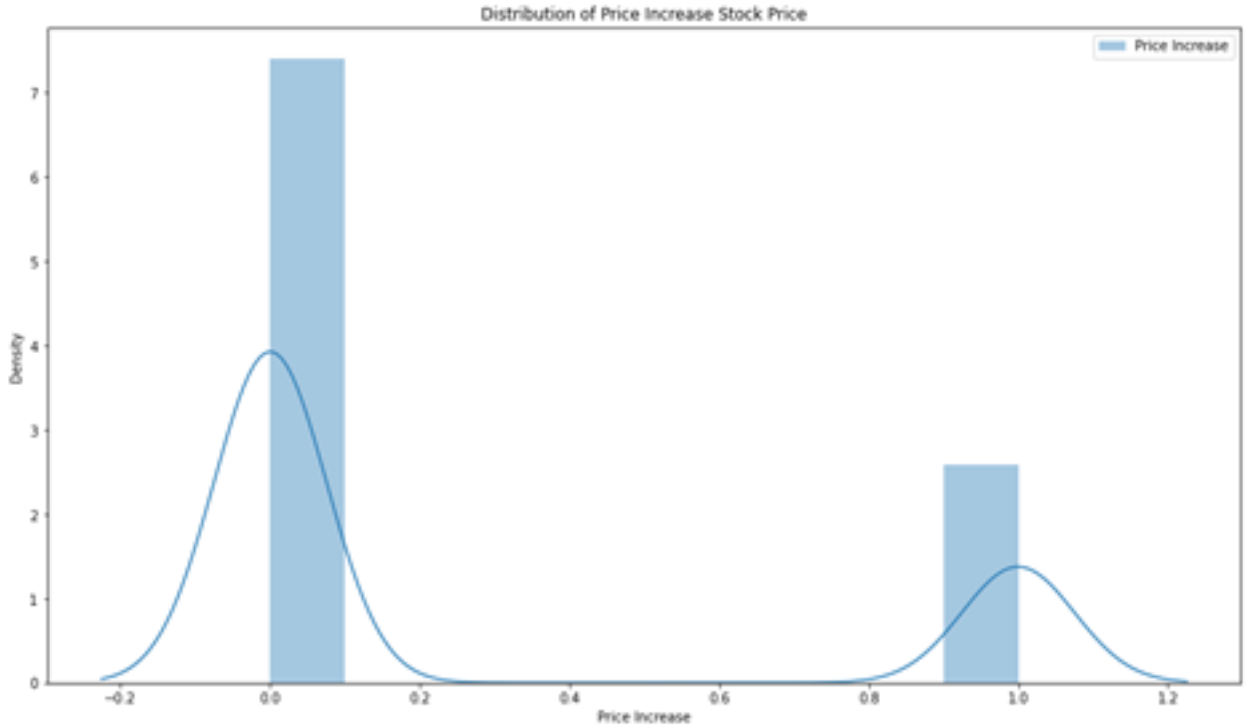
- **Histograms of the New Features**

```
In [47]: #Create some new features from the existing variables
selected_data["Day Price to Previous Ratio"] = selected_data["Day Price"] / selected_data["Previous"]
selected_data["Day High to Day Low Difference"] = selected_data["Day High"] - selected_data["Day Low"]

#Plot some histograms of the new features
selected_data.hist(column=["Day Price to Previous Ratio", "Day High to Day Low Difference"], figsize=(8, 4))
plt.show()
```



- **Distribution of Price Increase in the dataset**



8 Preprocessing

Converted the date column to datetime format and set it as the index.

```
In [23]: selected_data['Date'] = pd.to_datetime(selected_data['Date'])
         selected_data.set_index('Date', inplace=True)
```

```
In [24]: selected_data
```

Out[24]:

Date	Name	12m Low	12m High	Day Low	Day High	Day Price	Previous	Change	Volume	Price Increase
2022-01-03	Eaagads Ltd	10.00	15.00	13.50	13.80	13.50	13.50	-0.01	4000.0	0
2022-01-03	Kakuzi Plc	355.00	427.00	385.00	385.00	385.00	385.00	-0.01	10800.0	0
2022-01-03	Kapchorua Tea Kenya Plc	80.00	101.00	99.50	99.50	99.50	95.50	4.00	100.0	1
2022-01-03	Limuru Tea Plc	260.00	360.00	320.00	320.00	320.00	320.00	-0.01	10800.0	0
2022-01-03	Sasini Plc	16.75	22.60	18.70	18.70	18.70	18.70	-0.01	10800.0	0
...	...	...	...	...	...	...	...	...	...	...
2022-05-31	Mumias Sugar Company Ltd	0.27	0.27	0.27	0.27	0.27	0.27	-0.01	10800.0	0
2022-05-31	Unga Group Ltd	26.10	36.40	29.00	29.00	29.00	30.00	-1.00	2100.0	0
2022-05-31	Safaricom Plc	25.50	45.25	25.95	26.45	26.00	26.25	-0.25	20079900.0	0

Performed feature engineering to extract the day of the week, month, and year.

Feature Engineering :

Extract day of the week,month and year in the data set :

```
In [28]: selected_data['day_of_week'] = selected_data.index.dayofweek

         selected_data['month'] = selected_data.index.month

         selected_data['year'] = selected_data.index.year
```

9 Model Building

9.1 Splitting and Scaling the Dataset

We split the dataset into a training set and a test set with an 80-20 ratio. The objective was to use the training data to build classification models and then evaluate their performance on the test data.

I did some standardize to enable all features to participate in the learning process with equal chance.

Splitting the dataset into train and test set :

In [42]:
```
X = selected_data.drop('Price Increase', axis=1)
y = selected_data['Price Increase']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Scaling the data :

In [43]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

9.2 Classification

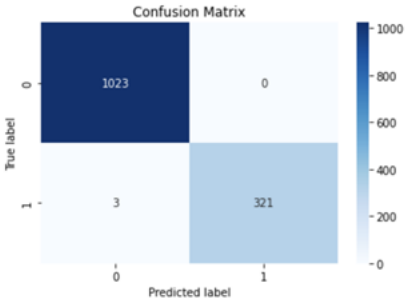
We framed the problem as a binary classification task, where the goal was to predict whether the 'Day Price' would increase or decrease compared to the previous day. We used three different classification models for this task:

9.2.1 Decision Tree Classifier

Trained the model on the training data, calculated and reported various classification metrics for both the test datasets, including accuracy, precision, recall, and F1-score.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1023
1	1.00	0.99	1.00	324
accuracy			1.00	1347
macro avg	1.00	1.00	1.00	1347
weighted avg	1.00	1.00	1.00	1347

Confusion Matrix for Decision

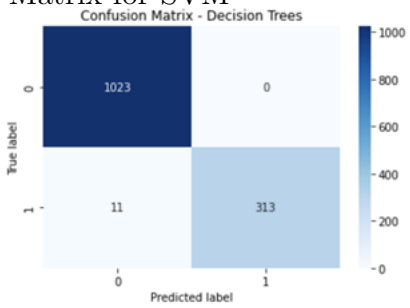


9.2.2 Support Vector Machines

Trained the model on the training data, calculated and reported various classification metrics for both the test datasets, including accuracy, precision, recall, and F1-score.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	1023
1	1.00	0.97	0.98	324
accuracy			0.99	1347
macro avg	0.99	0.98	0.99	1347
weighted avg	0.99	0.99	0.99	1347

Confusion Matrix for SVM

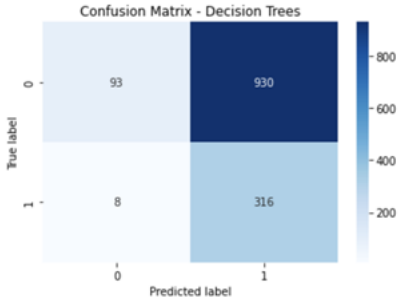


9.2.3 Gaussian Naive Bayes

Trained the model on the training data, calculated and reported various classification metrics for both the test datasets, including accuracy, precision, recall, and F1-score.

	precision	recall	f1-score	support
0	0.92	0.09	0.17	1023
1	0.25	0.98	0.40	324
accuracy			0.30	1347
macro avg	0.59	0.53	0.28	1347
weighted avg	0.76	0.30	0.22	1347

Confusion Matrix for Naive Bayes

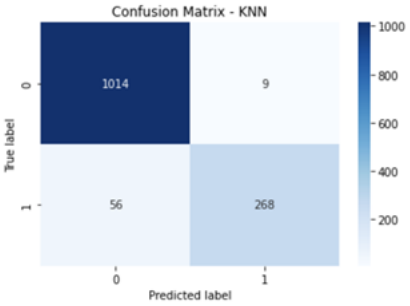


9.2.4 KNN

Trained the model on the training data, calculated and reported various classification metrics for both the test datasets, including accuracy, precision, recall, and F1-score.

	precision	recall	f1-score	support
0	0.95	0.99	0.97	1023
1	0.97	0.83	0.89	324
accuracy			0.95	1347
macro avg	0.96	0.91	0.93	1347
weighted avg	0.95	0.95	0.95	1347

Confusion Matrix for KNN



10 Comparison of Models

Metrics for each model (Accuracy, Precision, Recall, F1-score, AUC) were collected in a dataframe.

Out[93]:

	Model	Accuracy	Precision	Recall	F1-score	AUC
0	D1_Decision_Trees	0.997773	1.000000	0.990741	0.995349	0.995361
1	D1_SVM	0.991834	1.000000	0.966049	0.982732	0.992089
2	D1_Naive_Bayes	0.303638	0.253612	0.975309	0.402548	0.925866
3	D1_KNN	0.951745	0.967509	0.827160	0.891847	0.965669

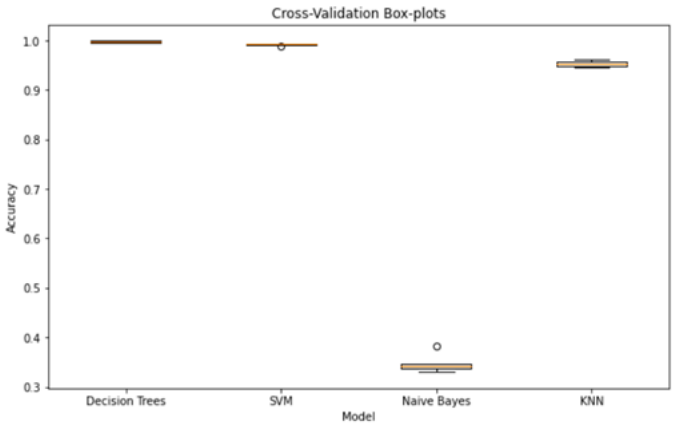
10.1 Cross-Validation Scores:

Cross-validation was used to assess model performance across folds, with average scores calculated for each model.

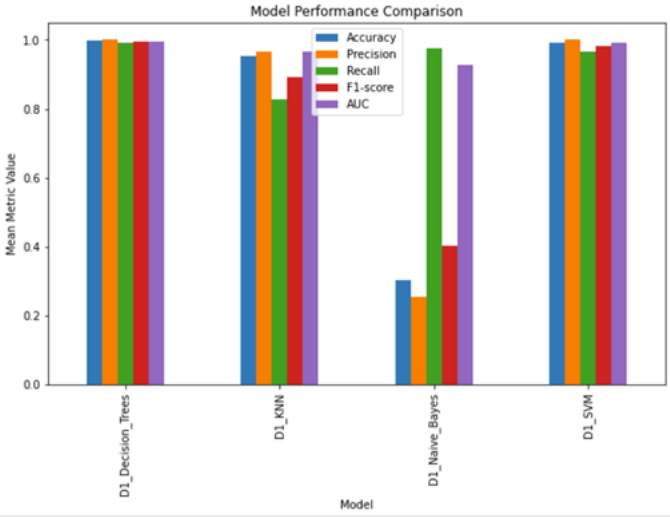
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Model	Average
0	0.999071	0.999071	0.996286	0.998143	0.996286	Decision Trees	0.997772
1	0.992572	0.991643	0.992572	0.992572	0.988858	SVM	0.991643
2	0.347261	0.330548	0.341690	0.382544	0.337047	Naive Bayes	0.347818
3	0.948932	0.945218	0.953575	0.957289	0.961931	KNN	0.953389

10.2 lot box plots of the cross-validation scores:

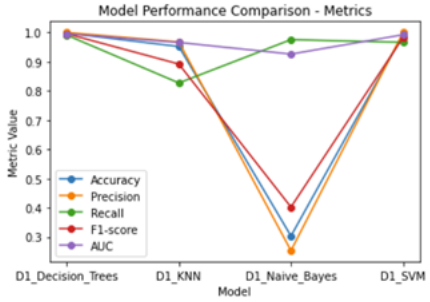
Box plots for cross-validation scores, a bar plot for comparing metric values, and ROC curves for all models were created.



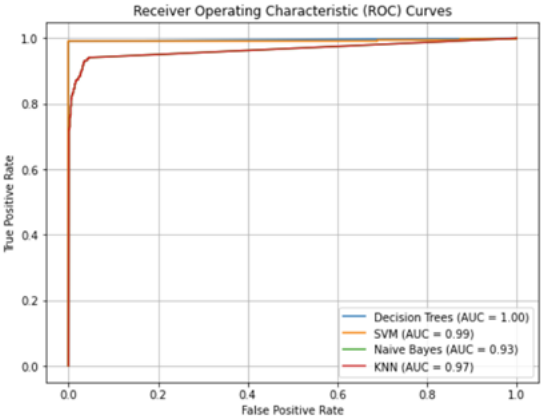
10.3 Plotting bar plot to compare the metric values of each model



10.4 Model Performance Comparison - Metrics



10.5 Receiver Operating Characteristic (ROC) Curves for all Models



11 Analysis of Results

The results we show above is the performance of three different machine learning models (Decision Trees, SVM, and Naive Bayes) on a binary classification problem.
The models are evaluated using six different metrics: accuracy, precision, recall, F1-score, and AUC as can been seen above.

A short analysis of the results is as follows:

- Decision Trees has the highest accuracy, F1-score, and recall, which means that it makes fewer mistakes overall, has a good balance between precision and recall, and is good at detecting true positives.

- SVM has the highest precision and AUC, which means that it avoids false positives very well, and has a good trade-off between the true positive rate and the false positive rate at different thresholds.

- Naive Bayes has the lowest accuracy, precision, F1-score, and AUC, which means that it makes many mistakes overall, produces many false positives, has a poor balance between precision and recall, and has a poor trade-off between the true positive rate and the false positive rate at different thresholds.

Based on these results, we can conclude that Decision Trees is the best model overall, while SVM is the best model for avoiding false positives, and Naive Bayes is the best model for detecting true positives. However, the choice of the best model may depend on the goal and the cost of the errors.