Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

In [3]:
```
df = pd.read_excel(r"C:\Users\AISWARYA\Downloads\FAANG.xlsx")
df.head()
```

Out[3]:

	Company	Ticker	Date	Open	High	Low	Close	Adj Close	Volume
0	Apple	AAPL	2005-01-03	1.156786	1.162679	1.117857	1.130179	0.954409	691992000
1	Apple	AAPL	2005-01-04	1.139107	1.169107	1.124464	1.141786	0.964210	1096810400
2	Apple	AAPL	2005-01-05	1.151071	1.165179	1.143750	1.151786	0.972655	680433600
3	Apple	AAPL	2005-01-06	1.154821	1.159107	1.130893	1.152679	0.973409	705555200
4	Apple	AAPL	2005-01-07	1.160714	1.243393	1.156250	1.236607	1.044284	2227450400

5 rows × 41 columns

In [5]:
```
df.shape
```

Out[5]: (23055, 41)

In [7]:
```
df.describe()
```

Out[7]:

	Date	Open	High	Low	Close	
count	23055	23055.000000	23055.000000	23055.000000	23055.000000	2305
mean	2015-05-26 16:14:03.201041152	93.647661	94.863101	92.420934	93.672274	9
min	2005-01-03 00:00:00	1.139107	1.159107	1.117857	1.130179	
25%	2010-09-22 12:00:00	11.728979	11.864486	11.587829	11.720929	1
50%	2015-08-21 00:00:00	38.584999	38.983002	38.297501	38.598499	3
75%	2020-03-20 00:00:00	134.849998	136.550003	133.449997	134.970001	13
max	2024-10-18 00:00:00	734.900024	736.000000	722.500000	730.289978	73
std	NaN	126.060231	127.749769	124.330704	126.069016	12

8 rows × 38 columns

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23055 entries, 0 to 23054
Data columns (total 41 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   Company                                23055 non-null  object
 1   Ticker                                 23055 non-null  object
 2   Date                                   23055 non-null  datetime64[ns]
 3   Open                                   23055 non-null  float64
 4   High                                   23055 non-null  float64
 5   Low                                    23055 non-null  float64
 6   Close                                  23055 non-null  float64
 7   Adj Close                              23055 non-null  float64
 8   Volume                                 23055 non-null  int64
 9   Market Cap                             23055 non-null  int64
 10  PE Ratio                               23055 non-null  float64
 11  Beta                                   18073 non-null  float64
 12  EPS                                    23055 non-null  float64
 13  Forward PE                             23055 non-null  float64
 14  Revenue                                0 non-null      float64
 15  Gross Profit                           0 non-null      float64
 16  Operating Income                       0 non-null      float64
 17  Net Income                             23055 non-null  int64
 18  Debt to Equity                         23055 non-null  float64
 19  Return on Equity (ROE)                 23055 non-null  float64
 20  Current Ratio                          23055 non-null  float64
 21  Dividends Paid                         13091 non-null  float64
 22  Dividend Yield                         13091 non-null  float64
 23  Quarterly Revenue Growth               23055 non-null  float64
 24  Analyst Recommendation                 23055 non-null  object
 25  Target Price                           23055 non-null  float64
 26  Free Cash Flow                         23055 non-null  int64
 27  Operating Margin                       23055 non-null  float64
 28  Profit Margin                          23055 non-null  float64
 29  Cash Ratio                             0 non-null      float64
 30  Quick Ratio                            23055 non-null  float64
 31  Price to Book Ratio                    23055 non-null  float64
 32  Enterprise Value                       23055 non-null  int64
 33  Total Debt                             23055 non-null  int64
 34  Total Assets                           0 non-null      float64
 35  Total Equity                           0 non-null      float64
 36  Beta (5Y)                              18073 non-null  float64
 37  Annual Dividend Rate                   13091 non-null  float64
 38  Trailing Twelve Months (TTM) Revenue   0 non-null      float64
 39  Trailing Twelve Months (TTM) EBITDA    0 non-null      float64
 40  Trailing Twelve Months (TTM) Earnings  0 non-null      float64
dtypes: datetime64[ns](1), float64(31), int64(6), object(3)
memory usage: 7.2+ MB
```

Exploratory Data Analysis

In [11]:

```
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Tesla Close price.', fontsize=15)
```

```
plt.ylabel('Price in dollars.')
plt.show()
```

In [13]:
```
df.head()
```

Out[13]:

	Company	Ticker	Date	Open	High	Low	Close	Adj Close	Volume
0	Apple	AAPL	2005-01-03	1.156786	1.162679	1.117857	1.130179	0.954409	691992000
1	Apple	AAPL	2005-01-04	1.139107	1.169107	1.124464	1.141786	0.964210	1096810400
2	Apple	AAPL	2005-01-05	1.151071	1.165179	1.143750	1.151786	0.972655	680433600
3	Apple	AAPL	2005-01-06	1.154821	1.159107	1.130893	1.152679	0.973409	705555200
4	Apple	AAPL	2005-01-07	1.160714	1.243393	1.156250	1.236607	1.044284	2227450400

5 rows × 41 columns

In [15]:
```
df[df['Close'] == df['Adj Close']].shape
```

Out[15]: (10068, 41)

In [17]:
```
df = df.drop(['Adj Close'], axis=1)
```

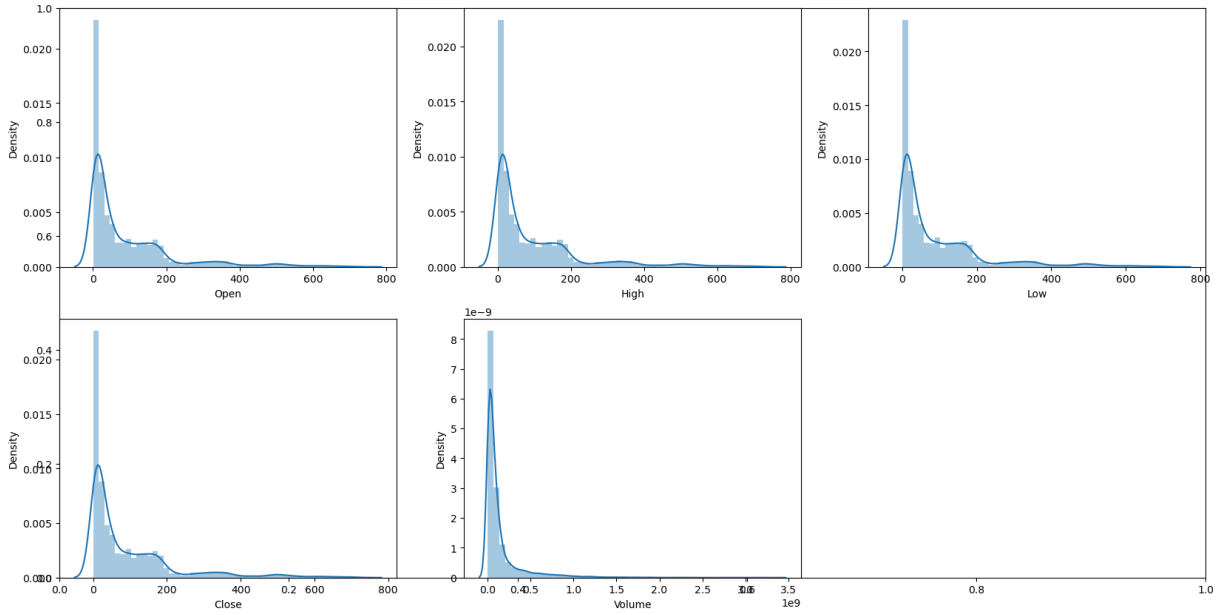
In [19]:
```
df.isnull().sum()
```

Out[19]:
```
Company                                0
Ticker                                 0
Date                                   0
Open                                   0
High                                   0
Low                                    0
Close                                  0
Volume                                 0
Market Cap                             0
PE Ratio                               0
Beta                                4982
EPS                                    0
Forward PE                             0
Revenue                            23055
Gross Profit                       23055
Operating Income                   23055
Net Income                             0
Debt to Equity                         0
Return on Equity (ROE)                 0
Current Ratio                          0
Dividends Paid                      9964
Dividend Yield                      9964
Quarterly Revenue Growth               0
Analyst Recommendation                 0
Target Price                           0
Free Cash Flow                         0
Operating Margin                       0
Profit Margin                          0
Cash Ratio                         23055
Quick Ratio                            0
Price to Book Ratio                    0
Enterprise Value                       0
Total Debt                             0
Total Assets                       23055
Total Equity                       23055
Beta (5Y)                           4982
Annual Dividend Rate                9964
Trailing Twelve Months (TTM) Revenue    23055
Trailing Twelve Months (TTM) EBITDA     23055
Trailing Twelve Months (TTM) Earnings   23055
dtype: int64
```

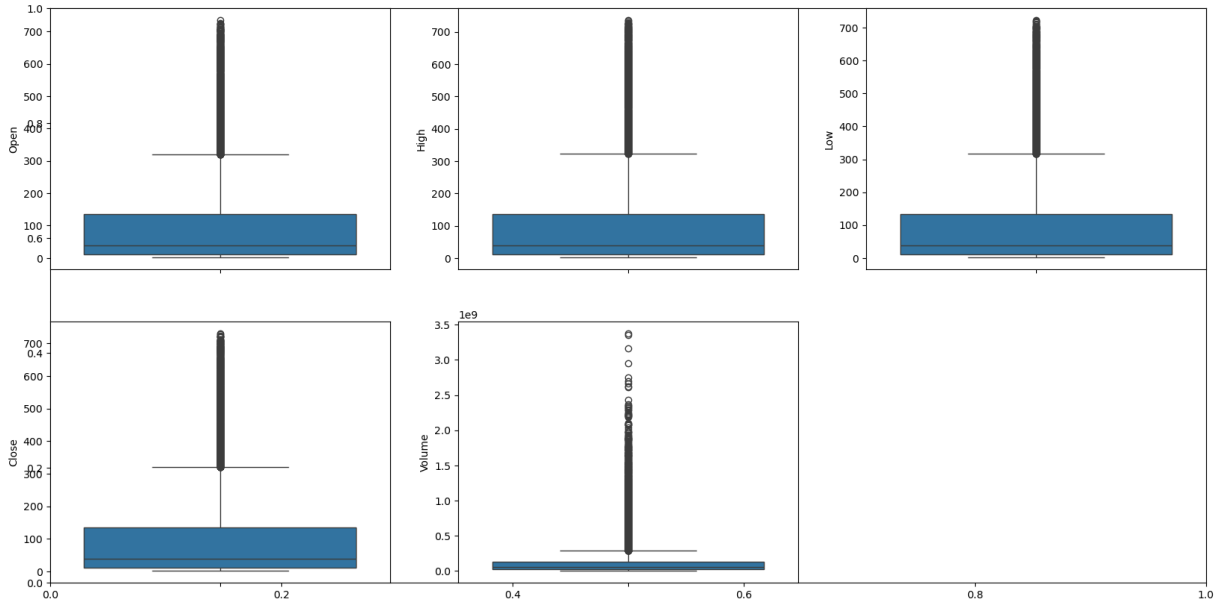
In [21]:
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']

plt.subplots(figsize=(20,10))

for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.distplot(df[col])
plt.show()
```



In [23]:
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.boxplot(df[col])
plt.show()
```



Feature Engineering

In [25]:
```
print(df['Date'].head())  # Inspect the first few rows of the 'Date' column
```

```
0   2005-01-03
1   2005-01-04
2   2005-01-05
3   2005-01-06
4   2005-01-07
Name: Date, dtype: datetime64[ns]
```

In [27]:
```
# Check if the 'Date' column is a string type
df['Date'] = df['Date'].astype(str)

# Split the 'Date' column by the '/' delimiter
splitted = df['Date'].str.split('/', expand=True)

# Ensure that the split operation returned exactly three parts (day, month,
if splitted.shape[1] == 3:
    df['month'] = splitted[0].astype(int)
    df['day'] = splitted[1].astype(int)
    df['year'] = splitted[2].astype(int)
else:
    print("Error: Date format is inconsistent.")

df.head()
```

```
Error: Date format is inconsistent.
```

Out[27]:

	Company	Ticker	Date	Open	High	Low	Close	Volume	Mar
0	Apple	AAPL	2005-01-03	1.156786	1.162679	1.117857	1.130179	691992000	3575090
1	Apple	AAPL	2005-01-04	1.139107	1.169107	1.124464	1.141786	1096810400	3575090
2	Apple	AAPL	2005-01-05	1.151071	1.165179	1.143750	1.151786	680433600	3575090
3	Apple	AAPL	2005-01-06	1.154821	1.159107	1.130893	1.152679	705555200	3575090
4	Apple	AAPL	2005-01-07	1.160714	1.243393	1.156250	1.236607	2227450400	3575090

5 rows × 40 columns

In [29]:
```
print(df.columns)
```

```
Index(['Company', 'Ticker', 'Date', 'Open', 'High', 'Low', 'Close', 'Volume',
       'Market Cap', 'PE Ratio', 'Beta', 'EPS', 'Forward PE', 'Revenue',
       'Gross Profit', 'Operating Income', 'Net Income', 'Debt to Equity',
       'Return on Equity (ROE)', 'Current Ratio', 'Dividends Paid',
       'Dividend Yield', 'Quarterly Revenue Growth', 'Analyst Recommendatio
n',
       'Target Price', 'Free Cash Flow', 'Operating Margin', 'Profit Margin',
       'Cash Ratio', 'Quick Ratio', 'Price to Book Ratio', 'Enterprise Valu
e',
       'Total Debt', 'Total Assets', 'Total Equity', 'Beta (5Y)',
       'Annual Dividend Rate', 'Trailing Twelve Months (TTM) Revenue',
       'Trailing Twelve Months (TTM) EBITDA',
       'Trailing Twelve Months (TTM) Earnings'],
      dtype='object')
```

In [31]:
```
print(df['Date'].head())  # Check the first few rows of the Date column
```

```
0    2005-01-03
1    2005-01-04
2    2005-01-05
3    2005-01-06
4    2005-01-07
Name: Date, dtype: object
```

In [33]:
```
import pandas as pd
import numpy as np

# Convert 'Date' column to datetime format if it's not already
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Check the conversion
print(df['Date'].head())  # Verify if the conversion was successful

# Extract the 'month' from the Date
df['month'] = df['Date'].dt.month

# Check if the 'month' column is created successfully
print(df['month'].head())

# Now, create the 'is_quarter_end' column based on whether the month is divi
df['is_quarter_end'] = np.where(df['month'] % 3 == 0, 1, 0)

# Display the updated dataframe
print(df.head())
```

```
0   2005-01-03
1   2005-01-04
2   2005-01-05
3   2005-01-06
4   2005-01-07
Name: Date, dtype: datetime64[ns]
0    1
1    1
2    1
3    1
4    1
Name: month, dtype: int32
  Company Ticker       Date      Open      High       Low     Close  \
0   Apple   AAPL 2005-01-03  1.156786  1.162679  1.117857  1.130179
1   Apple   AAPL 2005-01-04  1.139107  1.169107  1.124464  1.141786
2   Apple   AAPL 2005-01-05  1.151071  1.165179  1.143750  1.151786
3   Apple   AAPL 2005-01-06  1.154821  1.159107  1.130893  1.152679
4   Apple   AAPL 2005-01-07  1.160714  1.243393  1.156250  1.236607

       Volume     Market Cap   PE Ratio  ...    Total Debt  Total Assets  \
0   691992000  3575090000000  35.789955  ...  101304000000           NaN
1  1096810400  3575090000000  35.789955  ...  101304000000           NaN
2   680433600  3575090000000  35.789955  ...  101304000000           NaN
3   705555200  3575090000000  35.789955  ...  101304000000           NaN
4  2227450400  3575090000000  35.789955  ...  101304000000           NaN

   Total Equity  Beta (5Y)  Annual Dividend Rate  \
0           NaN      1.239                   1.0
1           NaN      1.239                   1.0
2           NaN      1.239                   1.0
3           NaN      1.239                   1.0
4           NaN      1.239                   1.0

   Trailing Twelve Months (TTM) Revenue  Trailing Twelve Months (TTM) EBITDA
\
0                                   NaN                                  NaN
1                                   NaN                                  NaN
2                                   NaN                                  NaN
3                                   NaN                                  NaN
4                                   NaN                                  NaN

   Trailing Twelve Months (TTM) Earnings  month  is_quarter_end
0                                    NaN      1              0
1                                    NaN      1              0
2                                    NaN      1              0
3                                    NaN      1              0
4                                    NaN      1              0

[5 rows x 42 columns]
```

In [35]:
```python
# Example: Fill NaT values with a default date or drop rows with NaT
df.dropna(subset=['Date'], inplace=True)  # Drop rows with invalid dates
# or
df['Date'].fillna('2000-01-01', inplace=True)  # Fill with a placeholder dat
```

In [37]:
```
df['is_quarter_end'] = np.where(df['month']%3==0,1,0)
df.head()
```

Out[37]:

	Company	Ticker	Date	Open	High	Low	Close	Volume	Mar
0	Apple	AAPL	2005-01-03	1.156786	1.162679	1.117857	1.130179	691992000	3575090
1	Apple	AAPL	2005-01-04	1.139107	1.169107	1.124464	1.141786	1096810400	3575090
2	Apple	AAPL	2005-01-05	1.151071	1.165179	1.143750	1.151786	680433600	3575090
3	Apple	AAPL	2005-01-06	1.154821	1.159107	1.130893	1.152679	705555200	3575090
4	Apple	AAPL	2005-01-07	1.160714	1.243393	1.156250	1.236607	2227450400	3575090

5 rows × 42 columns

In [39]:
```
print(df.head())  # Check first few rows to see the data structure
print(df.columns)  # List column names to ensure 'Date' is present
```

```
  Company Ticker       Date      Open      High       Low     Close  \
0   Apple   AAPL 2005-01-03  1.156786  1.162679  1.117857  1.130179
1   Apple   AAPL 2005-01-04  1.139107  1.169107  1.124464  1.141786
2   Apple   AAPL 2005-01-05  1.151071  1.165179  1.143750  1.151786
3   Apple   AAPL 2005-01-06  1.154821  1.159107  1.130893  1.152679
4   Apple   AAPL 2005-01-07  1.160714  1.243393  1.156250  1.236607

       Volume     Market Cap   PE Ratio  ...    Total Debt  Total Assets  \
0   691992000  3575090000000  35.789955  ...  101304000000           NaN
1  1096810400  3575090000000  35.789955  ...  101304000000           NaN
2   680433600  3575090000000  35.789955  ...  101304000000           NaN
3   705555200  3575090000000  35.789955  ...  101304000000           NaN
4  2227450400  3575090000000  35.789955  ...  101304000000           NaN

   Total Equity  Beta (5Y)  Annual Dividend Rate  \
0           NaN      1.239                   1.0
1           NaN      1.239                   1.0
2           NaN      1.239                   1.0
3           NaN      1.239                   1.0
4           NaN      1.239                   1.0

   Trailing Twelve Months (TTM) Revenue  Trailing Twelve Months (TTM) EBITDA
\
0                                   NaN                                  NaN
1                                   NaN                                  NaN
2                                   NaN                                  NaN
3                                   NaN                                  NaN
4                                   NaN                                  NaN

   Trailing Twelve Months (TTM) Earnings  month  is_quarter_end
0                                    NaN      1              0
1                                    NaN      1              0
2                                    NaN      1              0
3                                    NaN      1              0
4                                    NaN      1              0

[5 rows x 42 columns]
Index(['Company', 'Ticker', 'Date', 'Open', 'High', 'Low', 'Close', 'Volume',
       'Market Cap', 'PE Ratio', 'Beta', 'EPS', 'Forward PE', 'Revenue',
       'Gross Profit', 'Operating Income', 'Net Income', 'Debt to Equity',
       'Return on Equity (ROE)', 'Current Ratio', 'Dividends Paid',
       'Dividend Yield', 'Quarterly Revenue Growth', 'Analyst Recommendatio
n',
       'Target Price', 'Free Cash Flow', 'Operating Margin', 'Profit Margin',
       'Cash Ratio', 'Quick Ratio', 'Price to Book Ratio', 'Enterprise Valu
e',
       'Total Debt', 'Total Assets', 'Total Equity', 'Beta (5Y)',
       'Annual Dividend Rate', 'Trailing Twelve Months (TTM) Revenue',
       'Trailing Twelve Months (TTM) EBITDA',
       'Trailing Twelve Months (TTM) Earnings', 'month', 'is_quarter_end'],
      dtype='object')
```

In [41]:
```
print(df['Date'].isnull().sum())  # Check for missing values in 'Date'
```

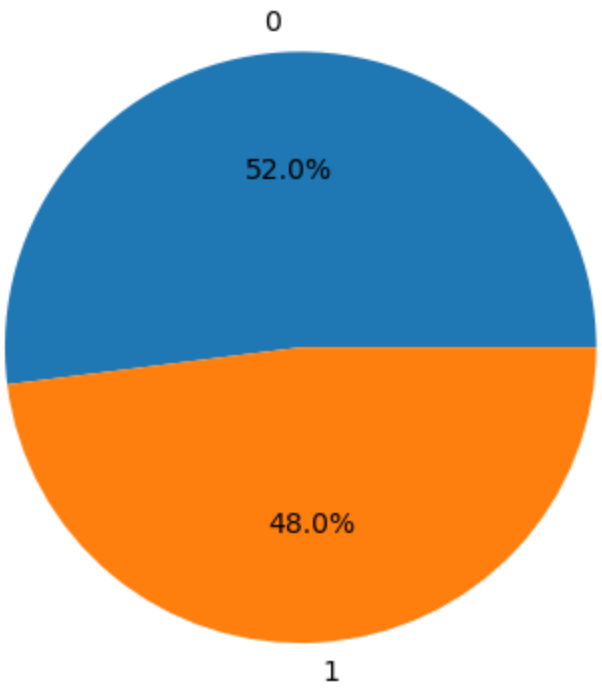
```
0
```

In [43]:
```
print(df['Date'].dtype)  # Should show 'datetime64[ns]'
```

```
datetime64[ns]
```

In [45]:
```
df['open-close']  = df['Open'] - df['Close']
df['low-high']  = df['Low'] - df['High']
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
```

In [47]:
```
plt.pie(df['target'].value_counts().values,
        labels=[0, 1], autopct='%1.1f%%')
plt.show()
```



Data Splitting and Normalization

In [49]:
```
features = df[['open-close', 'low-high', 'is_quarter_end']]
target = df['target']

scaler = StandardScaler()
features = scaler.fit_transform(features)

X_train, X_valid, Y_train, Y_valid = train_test_split(
    features, target, test_size=0.1, random_state=2022)
print(X_train.shape, X_valid.shape)
```

```
(20749, 3) (2306, 3)
```

Model Development and Evaluation

In [ ]:
```
models = [LogisticRegression(), SVC(
  kernel='poly', probability=True), XGBClassifier()]

for i in range(3):
```

```
  models[i].fit(X_train, Y_train)

  print(f'{models[i]} : ')
  print('Training Accuracy : ', metrics.roc_auc_score(
    Y_train, models[i].predict_proba(X_train)[:,1]))
  print('Validation Accuracy : ', metrics.roc_auc_score(
    Y_valid, models[i].predict_proba(X_valid)[:,1]))
  print()
```

```
LogisticRegression() :
Training Accuracy :  0.5129722165682219
Validation Accuracy :  0.5053046357167521
```

In [ ]:
```
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_estimator(models[0], X_valid, Y_valid)
plt.show()

# This code is modified by Susobhan Akhuli
```

In [ ]:
```

```