Stock Price Prediction Project

```
from platform import python_version
print(python_version())
```

```
3.9.12
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing Data

```
stock_df=pd.read_csv('Stock_Price_data_set.csv')
```

```
stock_df
```

```
            Date        Open        High         Low       Close   Adj
Close  \
0     2018-02-05  262.000000  267.899994  250.029999  254.259995
254.259995
1     2018-02-06  247.699997  266.700012  245.000000  265.720001
265.720001
2     2018-02-07  266.579987  272.450012  264.329987  264.559998
264.559998
3     2018-02-08  267.079987  267.619995  250.000000  250.100006
250.100006
4     2018-02-09  253.850006  255.800003  236.110001  249.470001
249.470001
...          ...         ...         ...         ...         ...
...
1004  2022-01-31  401.970001  427.700012  398.200012  427.140015
427.140015
1005  2022-02-01  432.959991  458.480011  425.540009  457.130005
457.130005
1006  2022-02-02  448.250000  451.980011  426.480011  429.480011
429.480011
1007  2022-02-03  421.440002  429.260010  404.279999  405.600006
405.600006
1008  2022-02-04  407.309998  412.769989  396.640015  410.170013
410.170013

        Volume
0     11896100
1     12595800
2      8981500
3      9306700
4     16906900
...        ...
1004  20047500
1005  22542300
1006  14346000
```

```
1007   9905200
1008   7782400

[1009 rows x 7 columns]
```

Exploring Data

```
stock_df.shape
```

```
(1009, 7)
```

```
stock_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1009 non-null   object 
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

```
stock_df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'],
dtype='object')
```

```
stock_df.describe
```

```
<bound method NDFrame.describe of             Date        Open        
High         Low       Close   Adj Close  \
0     2018-02-05  262.000000  267.899994  250.029999  254.259995  
254.259995   
1     2018-02-06  247.699997  266.700012  245.000000  265.720001  
265.720001   
2     2018-02-07  266.579987  272.450012  264.329987  264.559998  
264.559998   
3     2018-02-08  267.079987  267.619995  250.000000  250.100006  
250.100006   
4     2018-02-09  253.850006  255.800003  236.110001  249.470001  
249.470001   
...          ...         ...         ...         ...         ...  
...   
1004  2022-01-31  401.970001  427.700012  398.200012  427.140015  
427.140015   
1005  2022-02-01  432.959991  458.480011  425.540009  457.130005  
```

```
457.130005
1006  2022-02-02  448.250000  451.980011  426.480011  429.480011
429.480011
1007  2022-02-03  421.440002  429.260010  404.279999  405.600006
405.600006
1008  2022-02-04  407.309998  412.769989  396.640015  410.170013
410.170013

        Volume
0     11896100
1     12595800
2      8981500
3      9306700
4     16906900
...        ...
1004  20047500
1005  22542300
1006  14346000
1007   9905200
1008   7782400

[1009 rows x 7 columns]>
```

Missing Values

```
stock_df.isna().any()
```

```
Date         False
Open         False
High         False
Low          False
Close        False
Adj Close    False
Volume       False
dtype: bool
```

Duplicates

```
stock_df.duplicated().sum()
```

```
0
```

Column Data Type

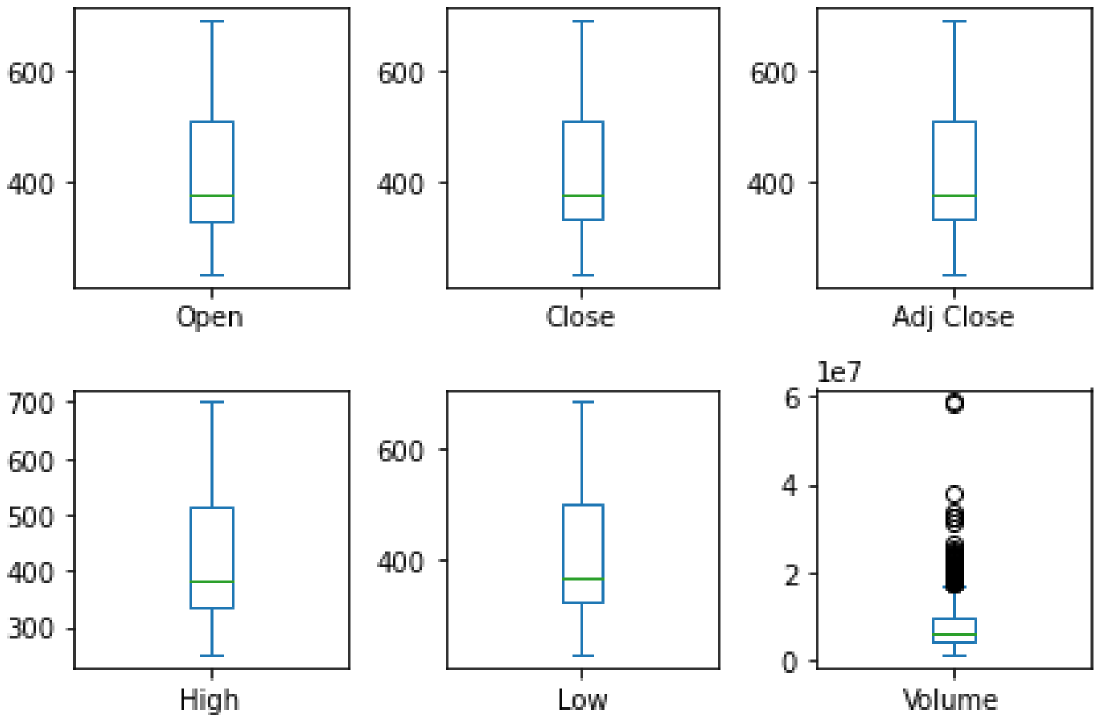
```
stock_df.dtypes
```

```
Date          object
Open         float64
High         float64
Low          float64
Close        float64
Adj Close    float64
```

```
Volume          int64
dtype: object
```

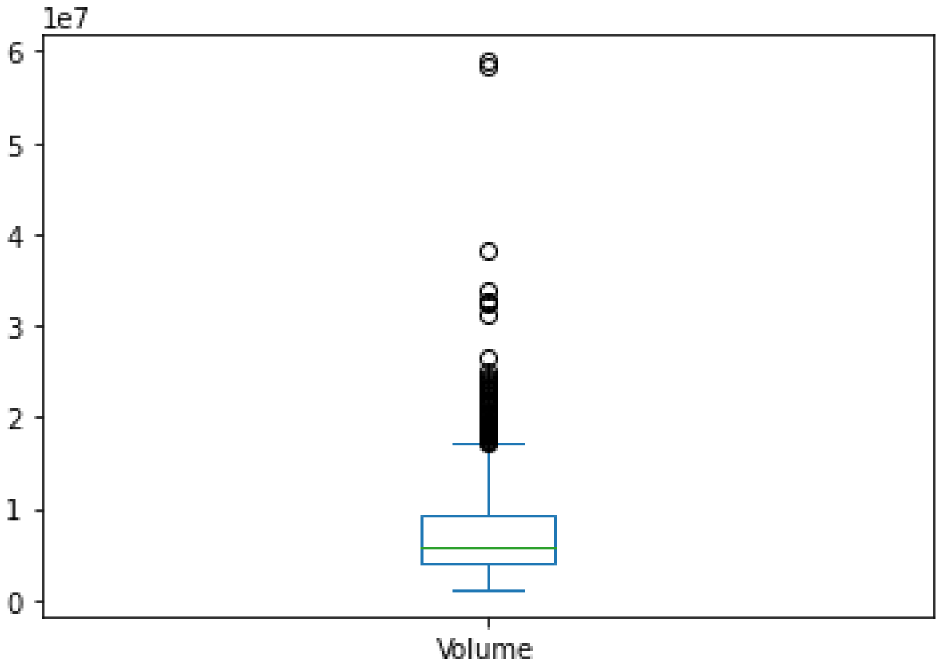
Outliers

```
plt.subplot(2,3,1)
stock_df['Open'].plot(kind='box')

plt.subplot(2,3,2)
stock_df['Close'].plot(kind='box')

plt.subplot(2,3,3)
stock_df['Adj Close'].plot(kind='box')

plt.subplot(2,3,4)
stock_df['High'].plot(kind='box')

plt.subplot(2,3,5)
stock_df['Low'].plot(kind='box')

plt.subplot(2,3,6)
stock_df['Volume'].plot(kind='box')

plt.tight_layout()
```



```
stock_df['Volume'].plot(kind='box')
```

```
<AxesSubplot:>
```



```
def find_outlier_limits(col_name):
    Q1,Q3=stock_df[col_name].quantile([.25,.75])
    IQR=Q3-Q1
    low=Q1-(2* IQR)
    high=Q3+(2* IQR)
    return (high,low)

high_vol,low_vol=find_outlier_limits('Volume')
print('Volume: ','upper limit: ',high_vol,' lower limit: ',low_vol)
```

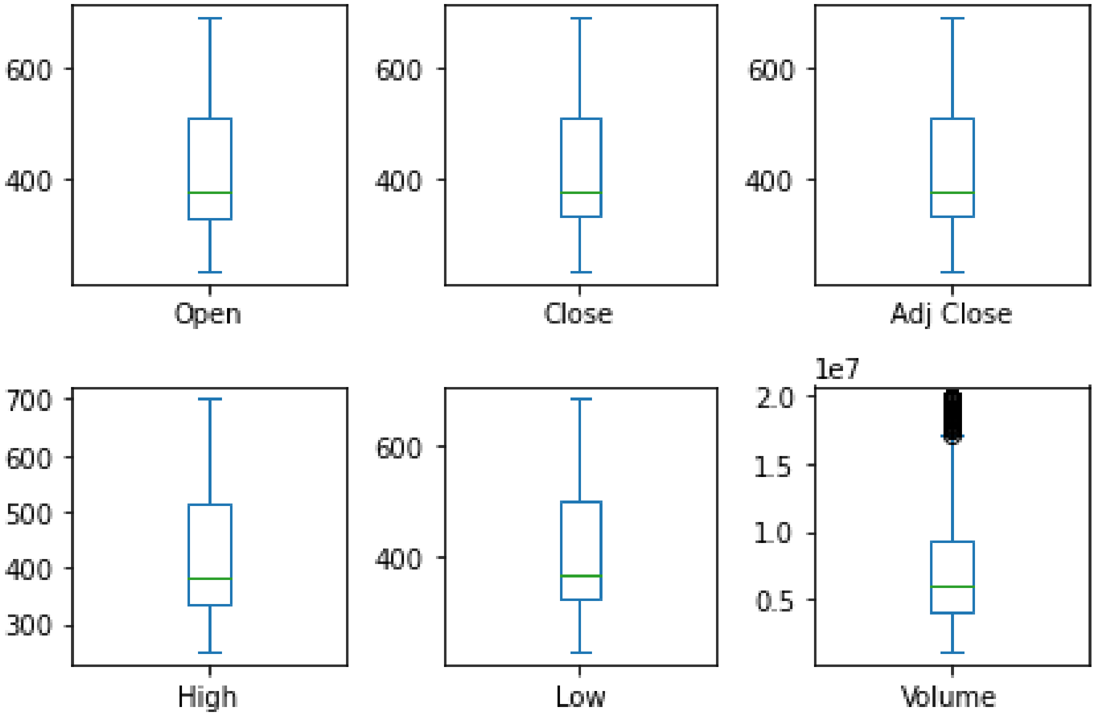
```
Volume:  upper limit:  19783400.0  lower limit:  -6369100.0
```

```
low_limit = 0
print('Volume: ','upper limit: ',high_vol,'lower limit: ',low_limit)
```

```
Volume:  upper limit:  19783400.0 lower limit:  0
```

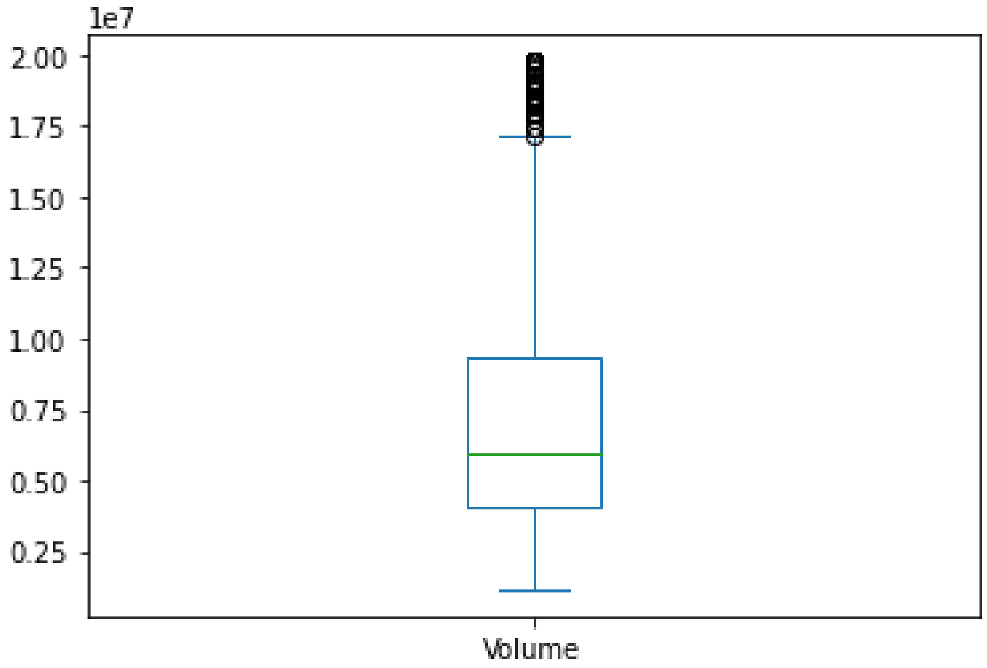
```
#replacing outliers value
stock_df.loc[stock_df['Volume'] > high_vol,'Volume'] = high_vol

stock_df.loc[stock_df['Volume']>high_vol,'Volume']=high_vol

plt.subplot(2,3,1)
stock_df['Open'].plot(kind='box')

plt.subplot(2,3,2)
stock_df['Close'].plot(kind='box')

plt.subplot(2,3,3)
stock_df['Adj Close'].plot(kind='box')
```

```
plt.subplot(2,3,4)
stock_df['High'].plot(kind='box')

plt.subplot(2,3,5)
stock_df['Low'].plot(kind='box')

plt.subplot(2,3,6)
stock_df['Volume'].plot(kind='box')

plt.tight_layout()
```



```
stock_df['Volume'].plot(kind='box')
```

```
<AxesSubplot:>
```



```
outliers = [stock_df['Volume'] > high_vol,'Volume']
outliers[True]
```

```
'Volume'
```

ML MODELING

```
stock_df
```

```
            Date        Open        High         Low       Close    Adj
Close  \
0     2018-02-05  262.000000  267.899994  250.029999  254.259995
254.259995
1     2018-02-06  247.699997  266.700012  245.000000  265.720001
265.720001
2     2018-02-07  266.579987  272.450012  264.329987  264.559998
264.559998
3     2018-02-08  267.079987  267.619995  250.000000  250.100006
250.100006
4     2018-02-09  253.850006  255.800003  236.110001  249.470001
249.470001
...          ...         ...         ...         ...         ...
...
1004  2022-01-31  401.970001  427.700012  398.200012  427.140015
427.140015
1005  2022-02-01  432.959991  458.480011  425.540009  457.130005
457.130005
1006  2022-02-02  448.250000  451.980011  426.480011  429.480011
429.480011
1007  2022-02-03  421.440002  429.260010  404.279999  405.600006
405.600006
```

```
1008  2022-02-04  407.309998  412.769989  396.640015  410.170013
410.170013

        Volume
0     11896100
1     12595800
2      8981500
3      9306700
4     16906900
...        ...
1004  19783400
1005  19783400
1006  14346000
1007   9905200
1008   7782400

[1009 rows x 7 columns]
```

```
X = stock_df.iloc[:, 1:8]
y = stock_df.iloc[:, 0]
print(X)
print(y)
```

```
            Open        High         Low       Close   Adj Close
Volume
0     262.000000  267.899994  250.029999  254.259995  254.259995
11896100
1     247.699997  266.700012  245.000000  265.720001  265.720001
12595800
2     266.579987  272.450012  264.329987  264.559998  264.559998
8981500
3     267.079987  267.619995  250.000000  250.100006  250.100006
9306700
4     253.850006  255.800003  236.110001  249.470001  249.470001
16906900
...          ...         ...         ...         ...         ...
...
1004  401.970001  427.700012  398.200012  427.140015  427.140015
19783400
1005  432.959991  458.480011  425.540009  457.130005  457.130005
19783400
1006  448.250000  451.980011  426.480011  429.480011  429.480011
14346000
1007  421.440002  429.260010  404.279999  405.600006  405.600006
9905200
1008  407.309998  412.769989  396.640015  410.170013  410.170013
7782400

[1009 rows x 6 columns]
0       2018-02-05
```

```
1       2018-02-06
2       2018-02-07
3       2018-02-08
4       2018-02-09
           ...
1004    2022-01-31
1005    2022-02-01
1006    2022-02-02
1007    2022-02-03
1008    2022-02-04
Name: Date, Length: 1009, dtype: object
```

```
X = pd.get_dummies(X)
X
```

```
            Open        High         Low       Close   Adj Close
Volume
0     262.000000  267.899994  250.029999  254.259995  254.259995
11896100
1     247.699997  266.700012  245.000000  265.720001  265.720001
12595800
2     266.579987  272.450012  264.329987  264.559998  264.559998
8981500
3     267.079987  267.619995  250.000000  250.100006  250.100006
9306700
4     253.850006  255.800003  236.110001  249.470001  249.470001
16906900
...          ...         ...         ...         ...         ...
...
1004  401.970001  427.700012  398.200012  427.140015  427.140015
19783400
1005  432.959991  458.480011  425.540009  457.130005  457.130005
19783400
1006  448.250000  451.980011  426.480011  429.480011  429.480011
14346000
1007  421.440002  429.260010  404.279999  405.600006  405.600006
9905200
1008  407.309998  412.769989  396.640015  410.170013  410.170013
7782400

[1009 rows x 6 columns]
```

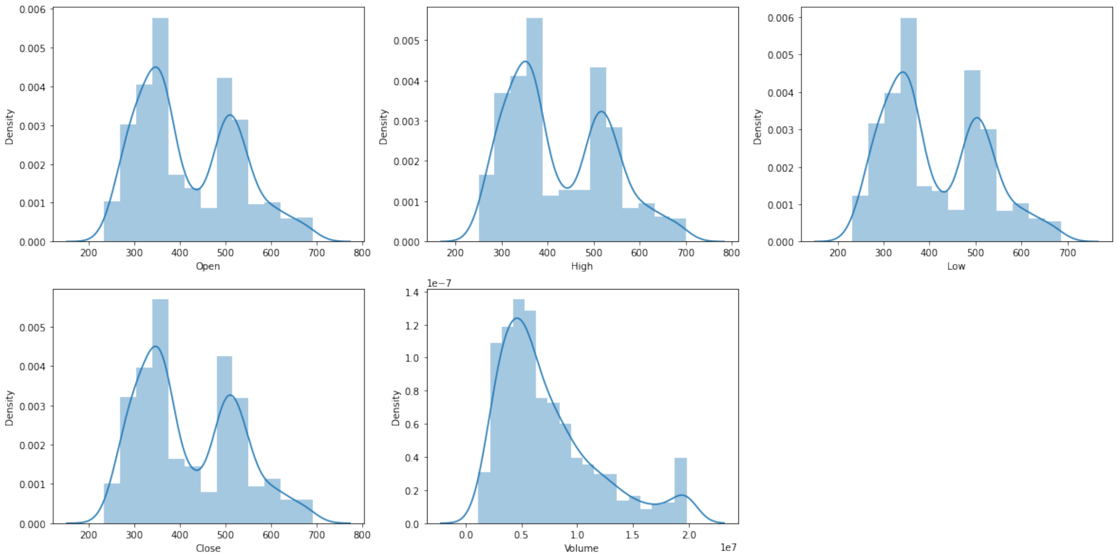
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
scale.fit_transform(X_train)
scale.transform(X_test);
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5,metric="euclidean")
```

TESTING

```
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')

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(stock_df[col])
plt.show()
```



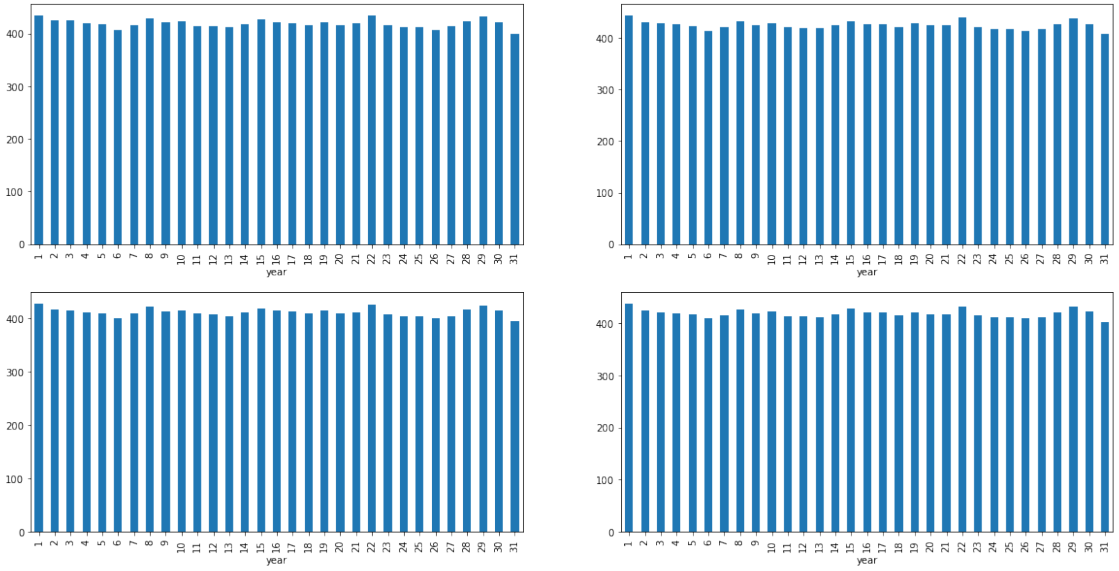
```
splitted = stock_df['Date'].str.split('-', expand=True)

stock_df['day'] = splitted[1].astype('int')
stock_df['month'] = splitted[0].astype('int')
stock_df['year'] = splitted[2].astype('int')

stock_df.head()
```

```
         Date         Open         High          Low        Close    Adj
Close  \
0  2018-02-05  262.000000  267.899994  250.029999  254.259995
254.259995
1  2018-02-06  247.699997  266.700012  245.000000  265.720001
265.720001
2  2018-02-07  266.579987  272.450012  264.329987  264.559998
264.559998
3  2018-02-08  267.079987  267.619995  250.000000  250.100006
250.100006
4  2018-02-09  253.850006  255.800003  236.110001  249.470001
249.470001

     Volume  day  month  year
0  11896100    2   2018     5
1  12595800    2   2018     6
2   8981500    2   2018     7
3   9306700    2   2018     8
4  16906900    2   2018     9
```

```
stock_df['is_quarter_end'] = np.where(stock_df['month']%3==0,1,0)
stock_df.head()
```

```
         Date        Open        High         Low       Close   Adj
Close  \
0  2018-02-05  262.000000  267.899994  250.029999  254.259995
254.259995
1  2018-02-06  247.699997  266.700012  245.000000  265.720001
265.720001
2  2018-02-07  266.579987  272.450012  264.329987  264.559998
264.559998
3  2018-02-08  267.079987  267.619995  250.000000  250.100006
250.100006
4  2018-02-09  253.850006  255.800003  236.110001  249.470001
249.470001

     Volume  day  month  year  is_quarter_end
0  11896100    2   2018     5               0
1  12595800    2   2018     6               0
2   8981500    2   2018     7               0
3   9306700    2   2018     8               0
4  16906900    2   2018     9               0
```

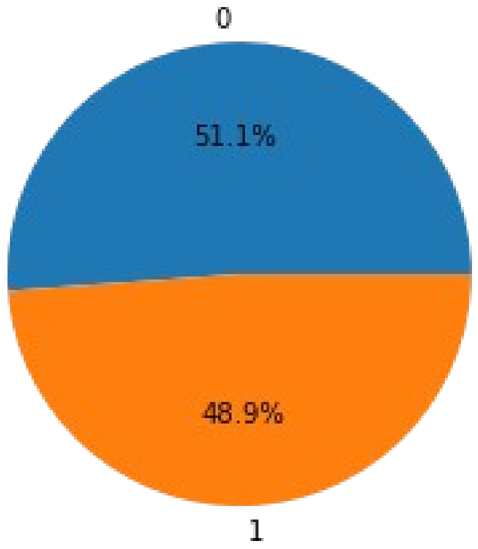
```
df=stock_df


data_grouped = df.groupby('year').mean()
plt.subplots(figsize=(20,10))

for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
  plt.subplot(2,2,i+1)
  data_grouped[col].plot.bar()
plt.show()
```



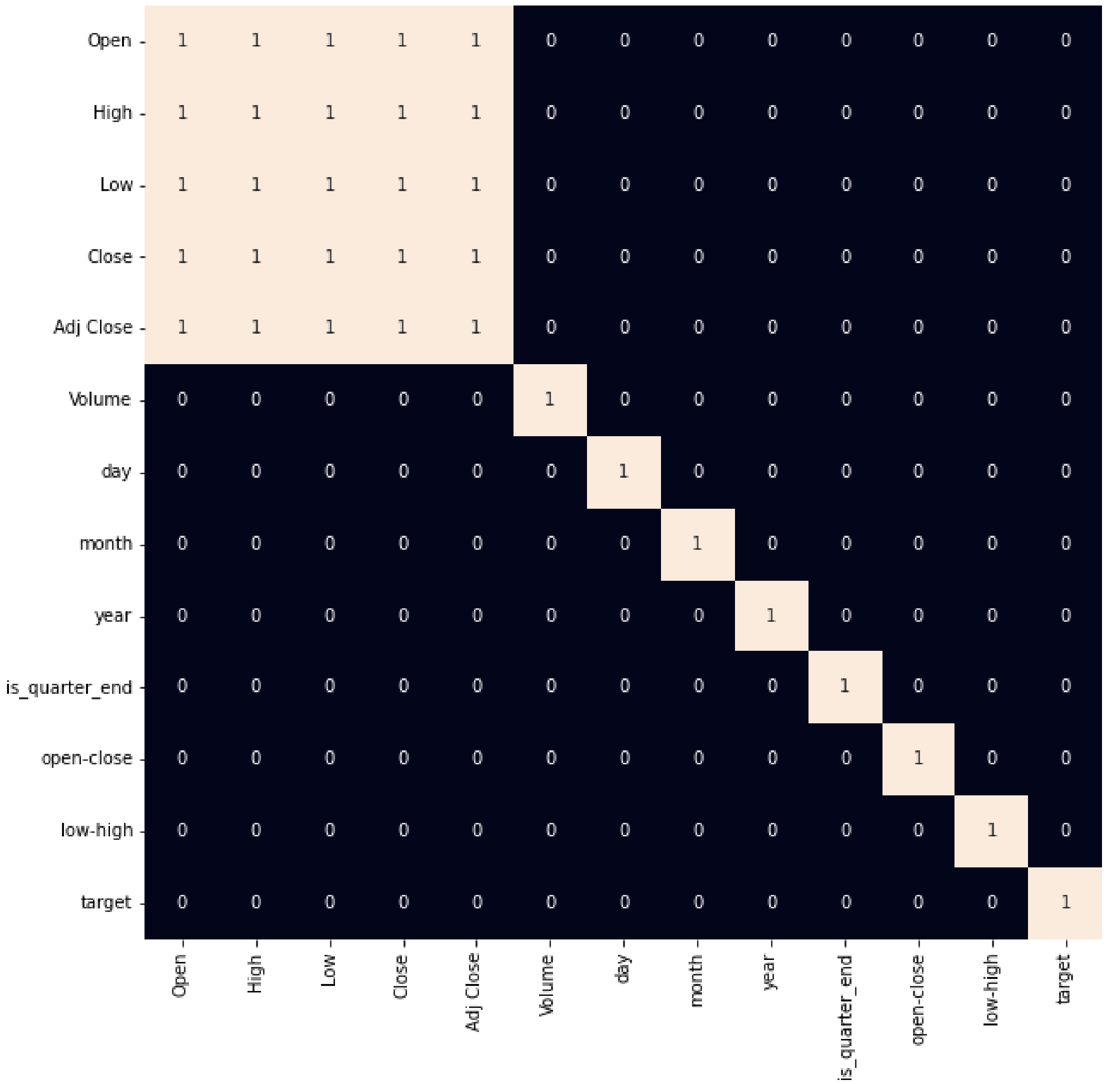
```
df.groupby('is_quarter_end').mean()
```

```
                      Open        High         Low       Close    Adj
Close  \
is_quarter_end

0               448.108608  454.981568  440.919864  448.071555
448.071555
1               341.911595  346.547464  336.562137  341.794528
341.794528

                      Volume       day        month       year
is_quarter_end
0               7.155014e+06  6.716235  2019.721692  15.785812
1               7.866719e+06  6.072464  2019.260870  15.536232
```

```
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)

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



```
plt.figure(figsize=(10, 10))
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
plt.show()
```



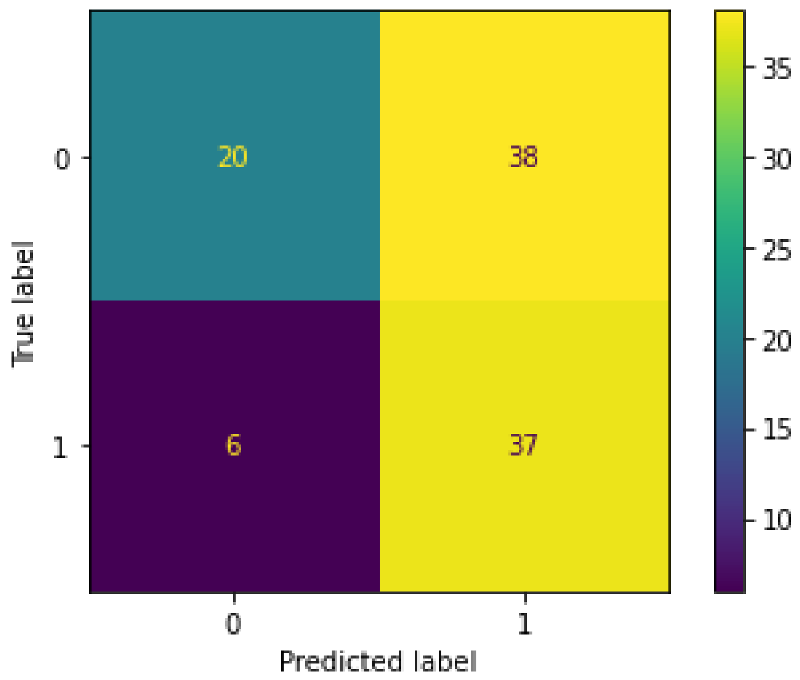
```
features = df[['open-close', 'low-high', 'is_quarter_end']]
target = df['target']

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)
```

```
(908, 3) (101, 3)
```

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

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 :  0.540628417292411
Validation Accuracy :  0.5725741780272654

SVC(kernel='poly', probability=True) :
Training Accuracy :  0.5316444314840465
Validation Accuracy :  0.6433440256615879


---------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
last)
Input In [39], in <cell line: 4>()
      1 models = [LogisticRegression(), SVC(
      2   kernel='poly', probability=True)]
      4 for i in range(3):
----> 5   models[i].fit(X_train, Y_train)
      7   print(f'{models[i]} : ')
      8   print('Training Accuracy : ', metrics.roc_auc_score(Y_train,
models[i].predict_proba(X_train)[:,1]))

IndexError: list index out of range
```

```
metrics.plot_confusion_matrix(models[0], X_valid, Y_valid)
plt.show()
```



Precesion of the code is not that good as it's only 37%.