In [1]:
```python
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')
```

In [2]:
```python
import yfinance as yf
```

In [3]:
```python
start = '2014-01-01'
end   = '2023-12-21'
stock = 'TSLA'

df = yf.download(stock, start , end)
df
```

```
[*********************100%%**********************]  1 of 1 completed
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2014-01-02	9.986667	10.165333	9.770000	10.006667	10.006667	92826000
2014-01-03	10.000000	10.146000	9.906667	9.970667	9.970667	70425000
2014-01-06	10.000000	10.026667	9.682667	9.800000	9.800000	80416500
2014-01-07	9.841333	10.026667	9.683333	9.957333	9.957333	75511500
2014-01-08	9.923333	10.246667	9.917333	10.085333	10.085333	92448000
...	...	...	...	...	...	...
2023-12-14	241.220001	253.880005	240.789993	251.050003	251.050003	160829200
2023-12-15	251.210007	254.130005	248.300003	253.500000	253.500000	135720800
2023-12-18	253.779999	258.739990	251.360001	252.080002	252.080002	116416500
2023-12-19	253.479996	258.339996	253.009995	257.220001	257.220001	106737400
2023-12-20	256.410004	259.839996	247.000000	247.139999	247.139999	125097000

2510 rows × 6 columns

In [4]:
```python
df.shape
```

Out[4]: (2510, 6)

In [5]:
```python
df.describe()
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
count	2510.000000	2510.000000	2510.000000	2510.000000	2510.000000	2.510000e+03
mean	93.709797	95.783245	91.482905	93.689099	93.689099	1.132260e+08
std	108.431664	110.868365	105.749029	108.345593	108.345593	7.556107e+07
min	9.366667	9.800000	9.111333	9.289333	9.289333	1.062000e+07
25%	15.751667	16.039166	15.478167	15.812667	15.812667	6.637762e+07
50%	21.722333	22.166334	21.425000	21.831000	21.831000	9.317100e+07
75%	199.274170	203.249996	193.885838	198.935833	198.935833	1.324391e+08
max	411.470001	414.496674	405.666656	409.970001	409.970001	9.140820e+08

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2510 entries, 2014-01-02 to 2023-12-20
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       2510 non-null   float64
 1   High       2510 non-null   float64
 2   Low        2510 non-null   float64
 3   Close      2510 non-null   float64
 4   Adj Close  2510 non-null   float64
 5   Volume     2510 non-null   int64  
dtypes: float64(5), int64(1)
memory usage: 137.3 KB
```

In [7]:
```
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Tesla Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()
```

In [8]:
```
df[df['Close'] == df['Adj Close']].shape
```

Out[8]:
```
(2510, 6)
```

In [9]:
```
df = df.drop(['Adj Close'], axis=1)
```

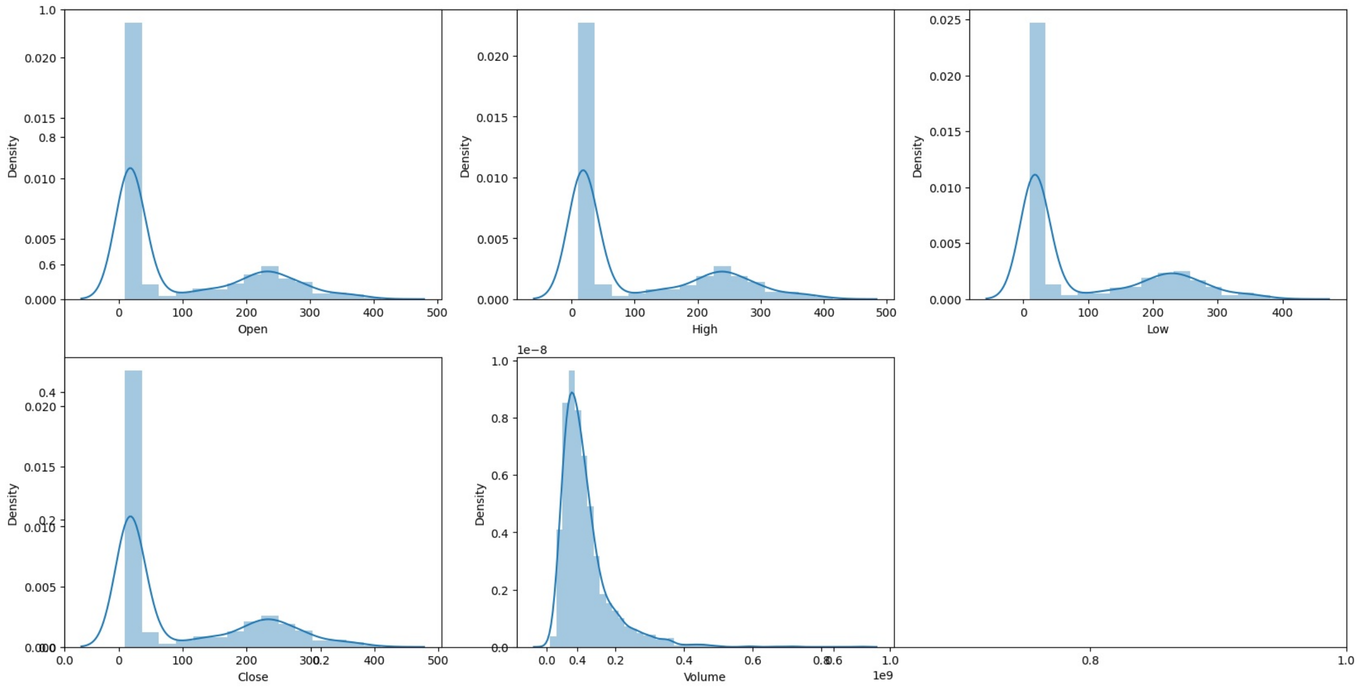
In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

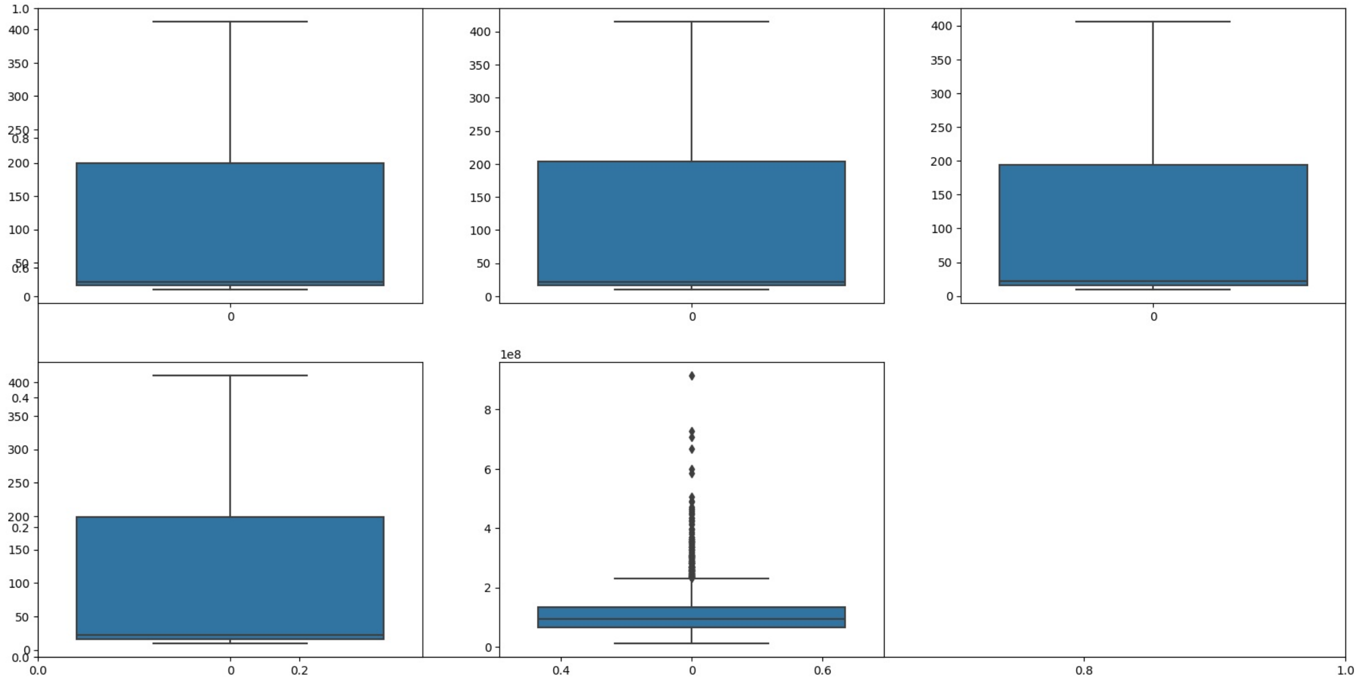
In [11]:
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']

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

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



In [12]:
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2,3,i+1)
    sb.boxplot(df[col])
plt.show()
```



In [13]:
```
df = df.reset_index()
```

In [14]:
```
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day
```

In [15]:
```
print(df.head())
```

```
         Date       Open       High       Low      Close    Volume  year  \
0  2014-01-02   9.986667  10.165333  9.770000  10.006667  92826000  2014
1  2014-01-03  10.000000  10.146000  9.906667   9.970667  70425000  2014
2  2014-01-06  10.000000  10.026667  9.682667   9.800000  80416500  2014
3  2014-01-07   9.841333  10.026667  9.683333   9.957333  75511500  2014
4  2014-01-08   9.923333  10.246667  9.917333  10.085333  92448000  2014

   month  day
0      1    2
1      1    3
2      1    6
3      1    7
4      1    8
```

In [16]:
```python
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day

df.head()
```

Out[16]:

	Date	Open	High	Low	Close	Volume	year	month	day
0	2014-01-02	9.986667	10.165333	9.770000	10.006667	92826000	2014	1	2
1	2014-01-03	10.000000	10.146000	9.906667	9.970667	70425000	2014	1	3
2	2014-01-06	10.000000	10.026667	9.682667	9.800000	80416500	2014	1	6
3	2014-01-07	9.841333	10.026667	9.683333	9.957333	75511500	2014	1	7
4	2014-01-08	9.923333	10.246667	9.917333	10.085333	92448000	2014	1	8

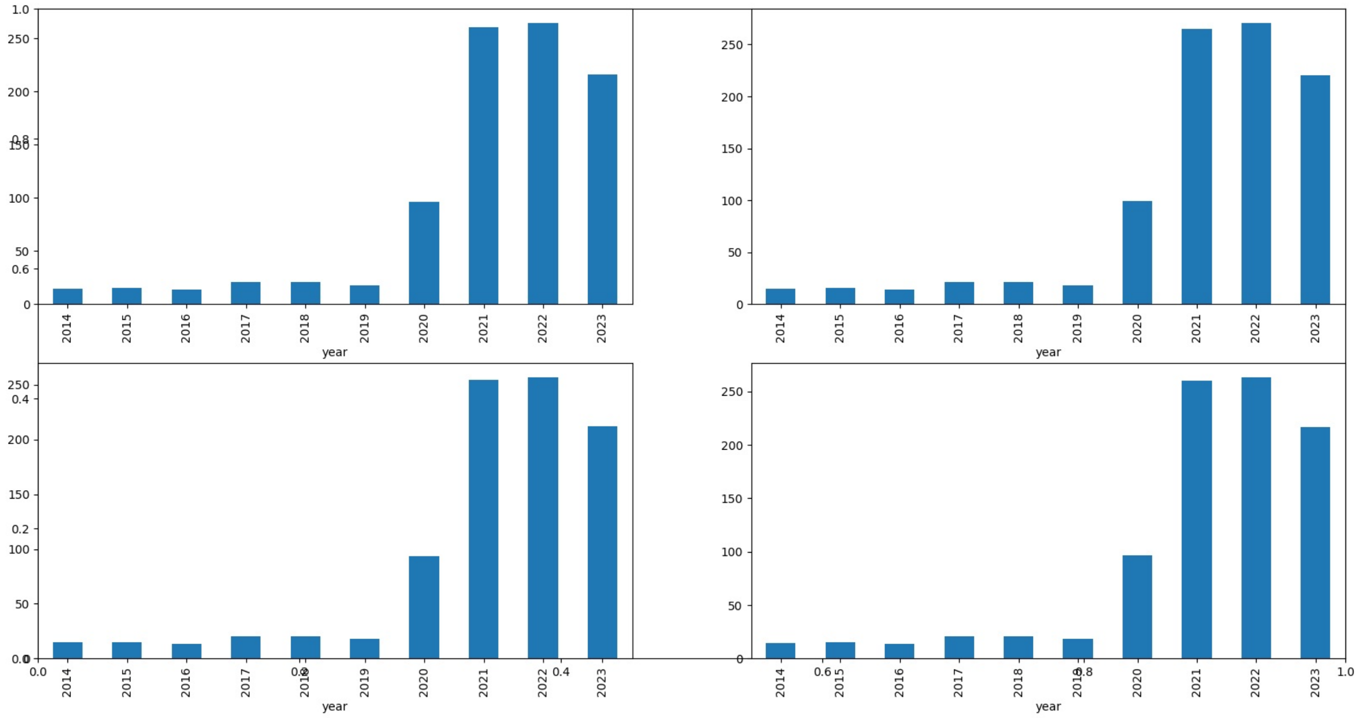
In [17]:
```python
df['is_quarter_end'] = np.where(df['month']%3==0,1,0)
df.head()
```

Out[17]:

	Date	Open	High	Low	Close	Volume	year	month	day	is_quarter_end
0	2014-01-02	9.986667	10.165333	9.770000	10.006667	92826000	2014	1	2	0
1	2014-01-03	10.000000	10.146000	9.906667	9.970667	70425000	2014	1	3	0
2	2014-01-06	10.000000	10.026667	9.682667	9.800000	80416500	2014	1	6	0
3	2014-01-07	9.841333	10.026667	9.683333	9.957333	75511500	2014	1	7	0
4	2014-01-08	9.923333	10.246667	9.917333	10.085333	92448000	2014	1	8	0

In [18]:
```python
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()
```



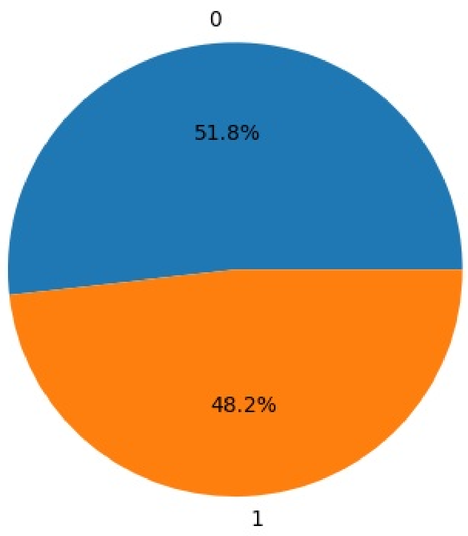
In [19]:
```python
df.groupby('is_quarter_end').mean()
```

Out[19]:

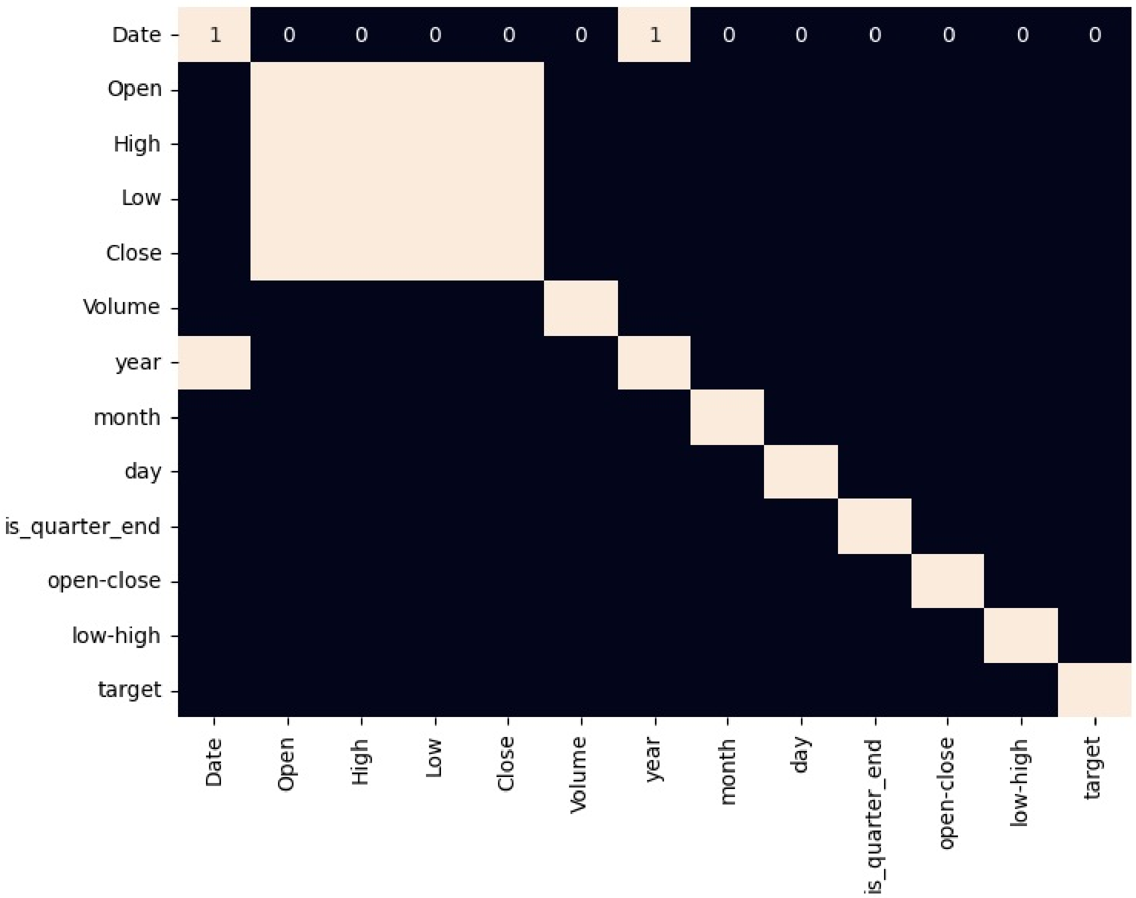
	Date	Open	High	Low	Close	Volume	year	month	day
is_quarter_end									
0	2018-12-16 12:50:07.923169024	92.942409	94.999027	90.710937	92.890010	1.142786e+08	2018.493998	6.098439	15.726891
1	2019-01-15 15:36:40.947867392	95.224571	97.331239	93.006718	95.266448	1.111482e+08	2018.469194	7.393365	15.708531

In [20]:
```python
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 [21]: plt.pie(df['target'].value_counts().values,
                labels=[0, 1], autopct='%1.1f%%')
         plt.show()
```




```
In [50]: plt.figure(figsize=(8, 6))
         sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
         plt.show()
```




```
In [23]: 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)
```

```
(2259, 3) (251, 3)
```

```
In [24]: models = [LogisticRegression(), SVC(
         kernel='poly', probability=True), XGBClassifier()]

         for i in range(3):
```

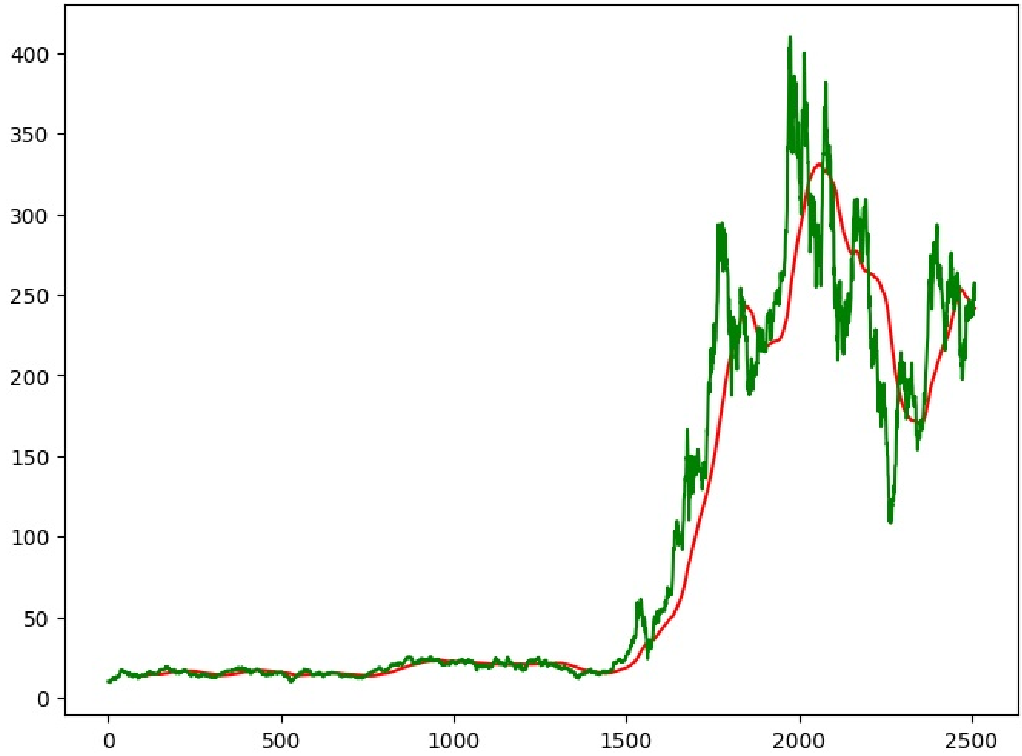
```python
    models[i].fit(X_train, Y_train)

  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()
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  0.9319439167825778
Validation Accuracy :  0.4924733231707317
```

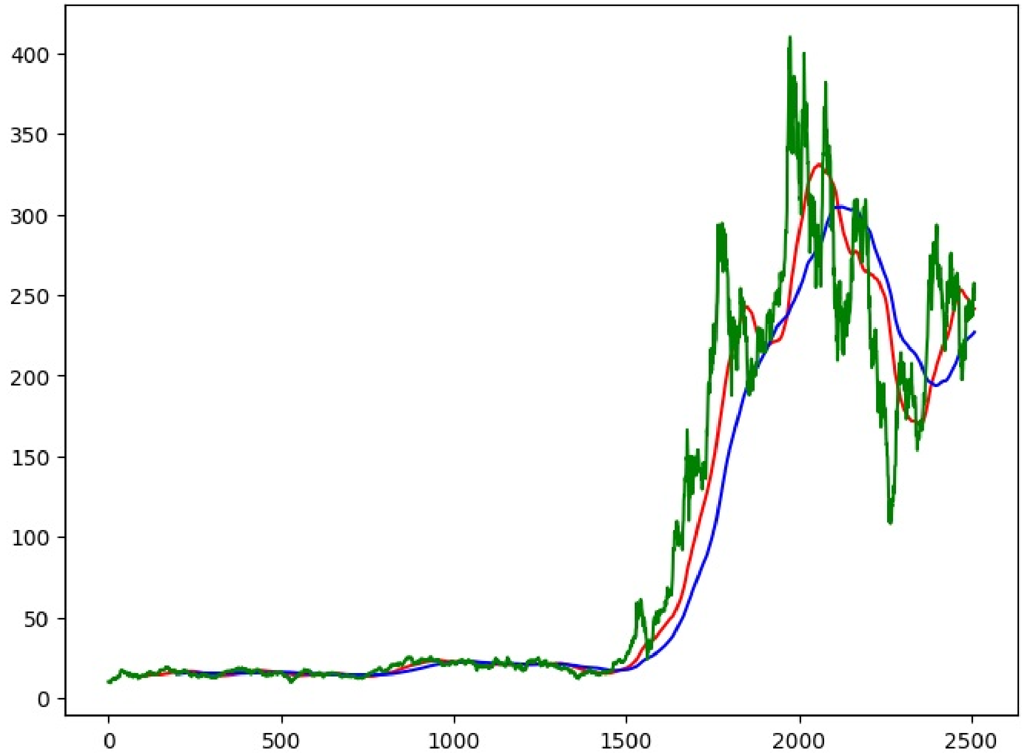
In [25]:
```python
ma_100_days = df.Close.rolling(100).mean()
```

In [26]:
```python
plt.figure(figsize=(8,6))
plt.plot(ma_100_days,'r')
plt.plot(df.Close , 'g')
plt.show()
```



In [27]:
```python
ma_200_days = df.Close.rolling(200).mean()
```

In [28]:
```python
plt.figure(figsize=(8,6))
plt.plot(ma_100_days,'r')
plt.plot(ma_200_days,'b')
plt.plot(df.Close , 'g')
plt.show()
```



In [29]:
```
data_train = pd.DataFrame(df.Close[0: int(len(df)*0.80)])
data_test = pd.DataFrame(df.Close[int(len(df)*0.80): len(df)])
```

In [30]:
```
data_train.shape[0]
```

Out[30]: 2008

In [31]:
```
data_test.shape[0]
```

Out[31]: 502

In [32]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
```

In [33]:
```
data_train_scale = scaler.fit_transform(data_train)
```

In [34]:
```
x = []
y = []

for i in range(100, data_train_scale.shape[0]):
    x.append(data_train_scale[i-100:i])
    y.append(data_train_scale[i,0])
```

In [35]:
```
x, y = np.array(x), np.array(y)
```

In [36]:
```
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
```

In [37]:
```
model = Sequential()
model.add(LSTM(units = 50 , activation = 'relu' , return_sequences = True,
              input_shape = ((x.shape[1],1))))

model.add(Dropout(0.2))

model.add(LSTM(units = 60, activation='relu' , return_sequences = True))
model.add(Dropout(0.3))

model.add(LSTM(units = 80 , activation='relu' , return_sequences = True))
model.add(Dropout(0.4))

model.add(LSTM(units = 120 , activation='relu' ))
model.add(Dropout(0.5))

model.add(Dense(units = 1))
```

In [38]:
```
model.compile(optimizer = 'adam' , loss = 'mean_squared_error')
```

In [39]:
```
model.fit(x,y, epochs = 50, batch_size = 32, verbose=1)
```

Epoch 1/50

```
60/60 [==============================] - 24s 298ms/step - loss: 0.0160
Epoch 2/50
60/60 [==============================] - 19s 316ms/step - loss: 0.0042
Epoch 3/50
60/60 [==============================] - 17s 281ms/step - loss: 0.0036
Epoch 4/50
60/60 [==============================] - 20s 341ms/step - loss: 0.0030
Epoch 5/50
60/60 [==============================] - 21s 343ms/step - loss: 0.0029
Epoch 6/50
60/60 [==============================] - 20s 337ms/step - loss: 0.0031
Epoch 7/50
60/60 [==============================] - 20s 341ms/step - loss: 0.0027
Epoch 8/50
60/60 [==============================] - 18s 304ms/step - loss: 0.0025
Epoch 9/50
60/60 [==============================] - 17s 285ms/step - loss: 0.0024
Epoch 10/50
60/60 [==============================] - 18s 307ms/step - loss: 0.0028
Epoch 11/50
60/60 [==============================] - 18s 297ms/step - loss: 0.0021
Epoch 12/50
60/60 [==============================] - 19s 308ms/step - loss: 0.0022
Epoch 13/50
60/60 [==============================] - 18s 295ms/step - loss: 0.0024
Epoch 14/50
60/60 [==============================] - 19s 311ms/step - loss: 0.0025
Epoch 15/50
60/60 [==============================] - 20s 328ms/step - loss: 0.0023
Epoch 16/50
60/60 [==============================] - 19s 324ms/step - loss: 0.0023
Epoch 17/50
60/60 [==============================] - 18s 302ms/step - loss: 0.0027
Epoch 18/50
60/60 [==============================] - 19s 322ms/step - loss: 0.0021
Epoch 19/50
60/60 [==============================] - 20s 329ms/step - loss: 0.0023
Epoch 20/50
60/60 [==============================] - 19s 314ms/step - loss: 0.0022
Epoch 21/50
60/60 [==============================] - 19s 313ms/step - loss: 0.0021
Epoch 22/50
60/60 [==============================] - 19s 313ms/step - loss: 0.0022
Epoch 23/50
60/60 [==============================] - 19s 322ms/step - loss: 0.0016
Epoch 24/50
60/60 [==============================] - 18s 306ms/step - loss: 0.0022
Epoch 25/50
60/60 [==============================] - 18s 301ms/step - loss: 0.0019
Epoch 26/50
60/60 [==============================] - 19s 316ms/step - loss: 0.0021
Epoch 27/50
60/60 [==============================] - 20s 335ms/step - loss: 0.0020
Epoch 28/50
60/60 [==============================] - 21s 349ms/step - loss: 0.0020
Epoch 29/50
60/60 [==============================] - 20s 329ms/step - loss: 0.0022
Epoch 30/50
60/60 [==============================] - 19s 313ms/step - loss: 0.0021
Epoch 31/50
60/60 [==============================] - 19s 311ms/step - loss: 0.0020
Epoch 32/50
60/60 [==============================] - 19s 321ms/step - loss: 0.0022
Epoch 33/50
60/60 [==============================] - 20s 327ms/step - loss: 0.0018
Epoch 34/50
60/60 [==============================] - 19s 313ms/step - loss: 0.0019
Epoch 35/50
60/60 [==============================] - 21s 344ms/step - loss: 0.0016
Epoch 36/50
60/60 [==============================] - 24s 398ms/step - loss: 0.0016
Epoch 37/50
60/60 [==============================] - 23s 389ms/step - loss: 0.0017
Epoch 38/50
60/60 [==============================] - 24s 399ms/step - loss: 0.0017
Epoch 39/50
60/60 [==============================] - 22s 373ms/step - loss: 0.0017
Epoch 40/50
60/60 [==============================] - 23s 383ms/step - loss: 0.0017
Epoch 41/50
60/60 [==============================] - 22s 374ms/step - loss: 0.0018
Epoch 42/50
60/60 [==============================] - 22s 370ms/step - loss: 0.0018
```

```
Epoch 43/50
60/60 [==============================] - 20s 331ms/step - loss: 0.0017
Epoch 44/50
60/60 [==============================] - 20s 333ms/step - loss: 0.0016
Epoch 45/50
60/60 [==============================] - 22s 366ms/step - loss: 0.0017
Epoch 46/50
60/60 [==============================] - 18s 298ms/step - loss: 0.0018
Epoch 47/50
60/60 [==============================] - 20s 341ms/step - loss: 0.0018
Epoch 48/50
60/60 [==============================] - 18s 301ms/step - loss: 0.0016
Epoch 49/50
60/60 [==============================] - 18s 307ms/step - loss: 0.0013
Epoch 50/50
60/60 [==============================] - 20s 329ms/step - loss: 0.0016
```

Out[39]: <keras.src.callbacks.History at 0x20076709950>

In [40]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 60)           26640

 dropout_1 (Dropout)         (None, 100, 60)           0

 lstm_2 (LSTM)               (None, 100, 80)           45120

 dropout_2 (Dropout)         (None, 100, 80)           0

 lstm_3 (LSTM)               (None, 120)               96480

 dropout_3 (Dropout)         (None, 120)               0

 dense (Dense)               (None, 1)                 121

=================================================================
Total params: 178761 (698.29 KB)
Trainable params: 178761 (698.29 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [41]:
```
pas_100_days = data_train.tail(100)
```

In [42]:
```
data_test = pd.concat([pas_100_days , data_test] , ignore_index = True)
data_test
```

Out[42]:

	Close
0	236.556671
1	236.580002
2	236.973328
3	238.210007
4	233.033340
...	...
597	251.050003
598	253.500000
599	252.080002
600	257.220001
601	247.139999

602 rows × 1 columns

In [43]:
```
data_test_scale = scaler.fit_transform(data_test)
```

In [44]:
```
x = []
y = []

for i in range(100, data_test_scale.shape[0]):
```

```
    x.append(data_test_scale[i-100:i])
    y.append(data_test_scale[i,0])

x , y = np.array(x) , np.array(y)
```

In [45]:
```
y_predict = model.predict(x)
```

```
16/16 [==============================] - 2s 55ms/step
```

In [46]:
```
scale = 1/scaler.scale_
```

In [47]:
```
y_predict = y_predict*scale
```

In [48]:
```
y = y*scale
```

In [49]:
```
plt.figure(figsize=(10,8))
plt.plot(y_predict, 'r', label = 'Predicted Price')
plt.plot(y, 'g', label = 'Original Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js