In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
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 sklearn import metrics
```

In [2]:
```
defaulter = pd.read_csv("defaulter.csv")
defaulter
```

Out[2]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
...	...	...	...	...	...	...	...
1004	31-01-2022	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	01-02-2022	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	02-02-2022	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	03-02-2022	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	04-02-2022	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

1009 rows × 7 columns

In [3]:
```
defaulter.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
```

In [11]:
```
x1 = defaulter.loc[0,["High","Volume"]]
x1
```

Out[11]:
```
High      267.899994
Volume     11896100
Name: 0, dtype: object
```

In [12]:
```
x2 = defaulter.loc[1,["High","Volume"]]
x2
```

Out[12]:
```
High      266.700012
Volume     12595800
Name: 1, dtype: object
```

In [13]:
```
np.linalg.norm(x1-x2)
```

Out[13]:
```
699700.000001029
```

In [19]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_values = scaler.fit_transform(defaulter[["High","Low"]])
defaulter["norm_High"] = scaled_values[:,0]
defaulter["norm_Low"] = scaled_values[:,0]
defaulter
```

Out[19]:

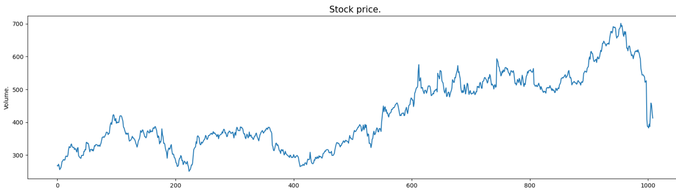
	Date	Open	High	Low	Close	Adj Close	Volume	norm_High	norm_Volume	norm_Low
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100	0.038304	0.038304	0.038304
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800	0.035640	0.035640	0.035640
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500	0.048408	0.048408	0.048408
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700	0.037683	0.037683	0.037683
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900	0.011436	0.011436	0.011436
...	...	...	...	...	...	...	...	...	...	...
1004	31-01-2022	401.970001	427.700012	398.200012	427.140015	427.140015	20047500	0.393147	0.393147	0.393147
1005	01-02-2022	432.959991	458.480011	425.540009	457.130005	457.130005	22542300	0.461496	0.461496	0.461496
1006	02-02-2022	448.250000	451.980011	426.480011	429.480011	429.480011	14346000	0.447062	0.447062	0.447062
1007	03-02-2022	421.440002	429.260010	404.279999	405.600006	405.600006	9905200	0.396611	0.396611	0.396611
1008	04-02-2022	407.309998	412.769989	396.640015	410.170013	410.170013	7782400	0.359995	0.359995	0.359995

1009 rows × 10 columns

In [28]:
```
defaulter.shape
```

Out[28]:
```
(1009, 10)
```

In [37]:
```
plt.figure(figsize=(20,5))
plt.plot(defaulter['High'])
plt.title('Stock price.', fontsize=15)
plt.ylabel('Volume.')
plt.show()
```



In [38]:
```
defaulter.head()
```

Out[38]:

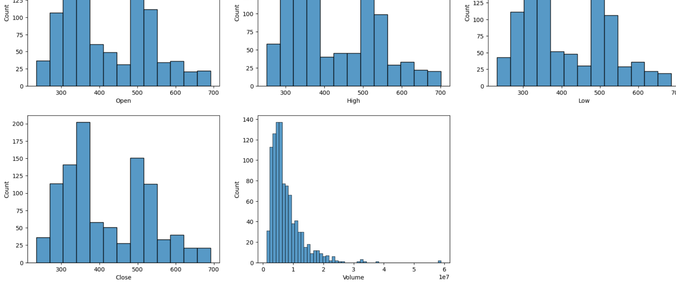
	Date	Open	High	Low	Close	Adj Close	Volume	norm_High	norm_Volume	norm_Low
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100	0.038304	0.038304	0.038304
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800	0.035640	0.035640	0.035640
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500	0.048408	0.048408	0.048408
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700	0.037683	0.037683	0.037683
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900	0.011436	0.011436	0.011436

In [42]:
```
defaulter.isnull().sum()
```

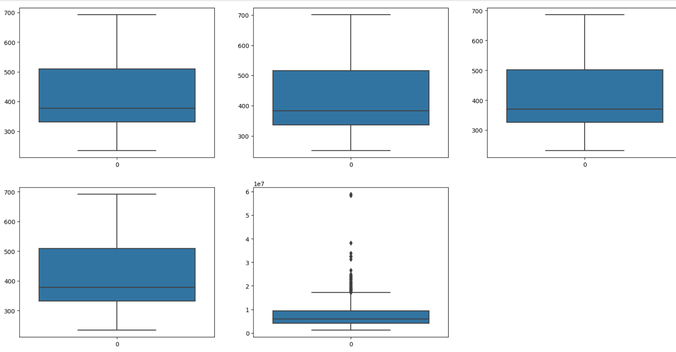
Out[42]:
```
Date          0
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
norm_High     0
norm_Volume   0
norm_Low      0
dtype: int64
```

In [46]:
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2,3,i+1)
    sb.histplot(defaulter[col])
plt.show()
```

```
C:\Users\rames_ykh6ekx\AppData\Local\Temp\ipykernel_18052\48507062.py:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(2,3,i+1)
```



In [50]:
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.boxplot(defaulter[col])
plt.show()
```



In [53]:
```
defaulter['is_quarter_end'] = np.where(defaulter['Volume']%3==0,1,0)
defaulter.head()
```

Out[53]:

	Date	Open	High	Low	Close	Adj Close	Volume	norm_High	norm_Volume	norm_Low	is_quarter_end
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100	0.038304	0.038304	0.038304	0
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800	0.035640	0.035640	0.035640	1
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500	0.048408	0.048408	0.048408	0
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700	0.037683	0.037683	0.037683	0
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900	0.011436	0.011436	0.011436	0