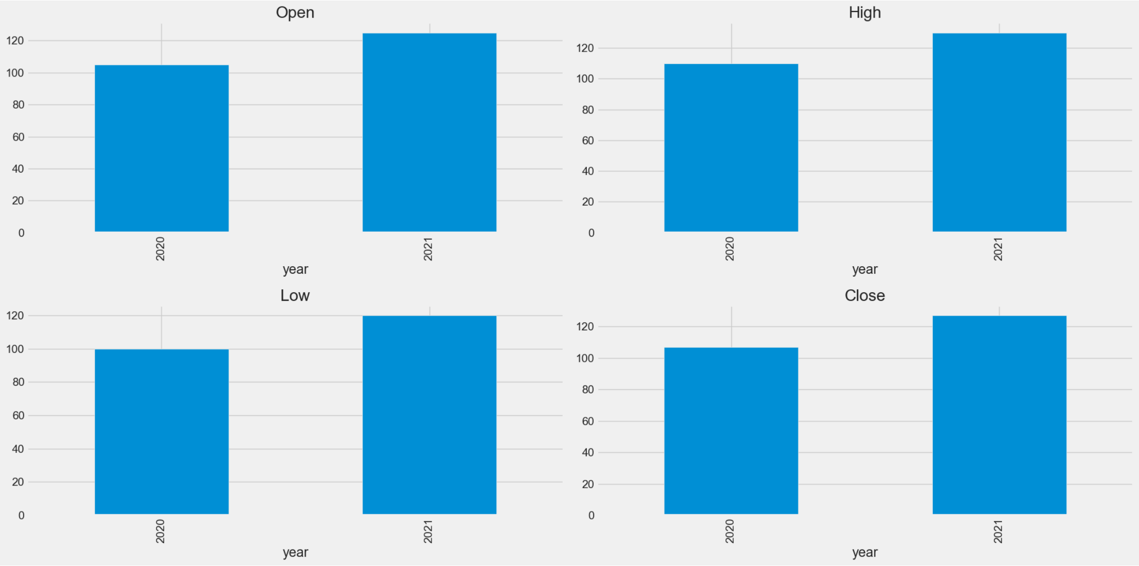
In [130]:

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

# Sample DataFrame
data = {'Date': ['2020-01-01', '2020-02-01', '2021-01-01', '2021-02-01'],
        'Open': [100, 110, 120, 130],
        'High': [105, 115, 125, 135],
        'Low': [95, 105, 115, 125],
        'Close': [102, 112, 122, 132]}

df = pd.DataFrame(data)

# Extract the year from the 'Date' column
df['year'] = pd.to_datetime(df['Date']).dt.year

# Now you can group by 'year'
data_grouped = df.groupby('year').mean()

# Create four subplots, one for each column
plt.figure(figsize=(20, 10))
for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
    plt.subplot(2, 2, i + 1)
    data_grouped[col].plot(kind='bar', title=col)

plt.tight_layout()
plt.show()

from sklearn import metrics

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



In [131]:

```
df = pd.read_csv('TSLA.csv')
df.head()
```

Out[131]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	2010-06-30	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	2010-07-01	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	2010-07-02	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	2010-07-06	20.000000	20.00	15.830000	16.110001	16.110001	6866900

In [132]:

```
df.shape
```

Out[132]:

```
(2416, 7)
```

In [133]:

```
df.describe()
```

Out[133]:

	Open	High	Low	Close	Adj Close	Volume
count	2416.000000	2416.000000	2416.000000	2416.000000	2416.000000	2.416000e+03
mean	186.271147	189.578224	182.916639	186.403651	186.403651	5.572722e+06
std	118.740163	120.892329	116.857591	119.136020	119.136020	4.987809e+06
min	16.139999	16.629999	14.980000	15.800000	15.800000	1.185000e+05
25%	34.342498	34.897501	33.587501	34.400002	34.400002	1.899275e+06
50%	213.035004	216.745002	208.870002	212.960007	212.960007	4.578400e+06
75%	266.450012	270.927513	262.102501	266.774994	266.774994	7.361150e+06
max	673.690002	786.140015	673.520020	780.000000	780.000000	4.706500e+07

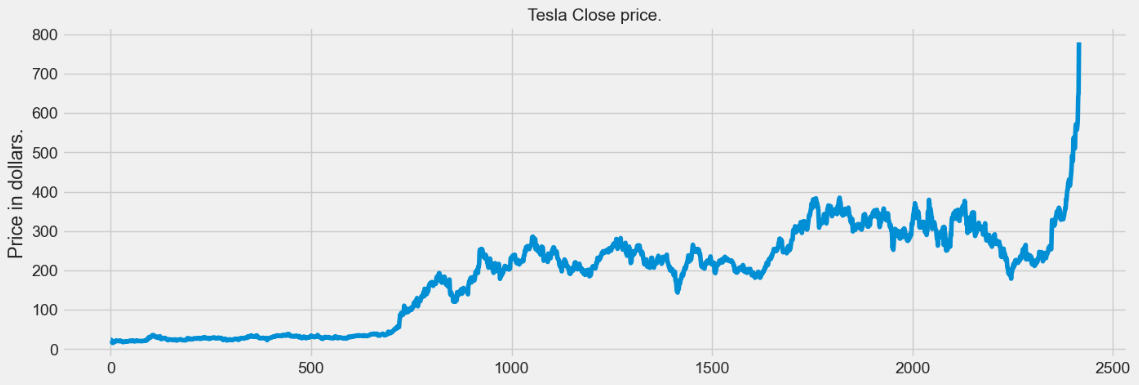
In [119]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2416 entries, 0 to 2415
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2416 non-null   object 
 1   Open       2416 non-null   float64
 2   High       2416 non-null   float64
 3   Low        2416 non-null   float64
 4   Close      2416 non-null   float64
 5   Adj Close  2416 non-null   float64
 6   Volume     2416 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 132.3+ KB
```

In [120]:

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



In [121]:

```
df.head()
```

Out[121]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	2010-06-30	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	2010-07-01	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	2010-07-02	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	2010-07-06	20.000000	20.00	15.830000	16.110001	16.110001	6866900

In [134]:

```
df[df['Close'] == df['Adj Close']].shape
```

Out[134]:

```
(2416, 7)
```

In [135]:

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

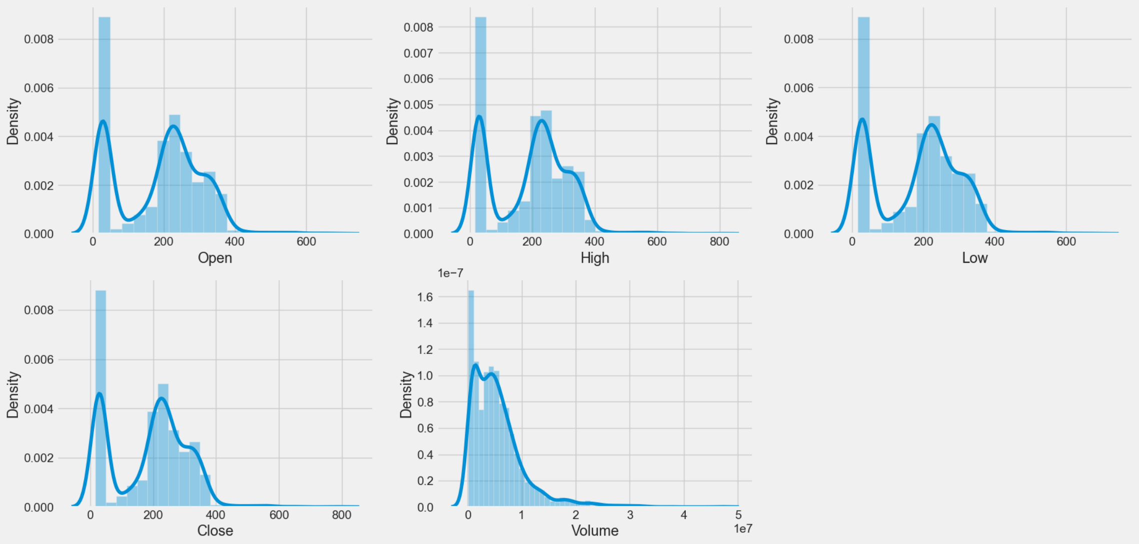
In [124]:

```
df.isnull().sum()
```

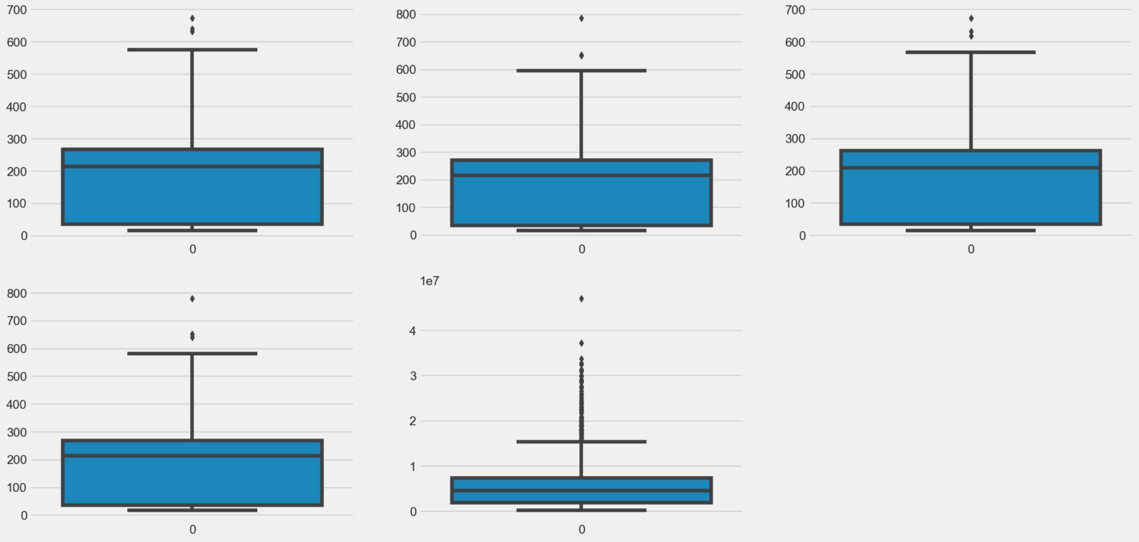
Out[124]:

```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

In [125]:
```
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()
```

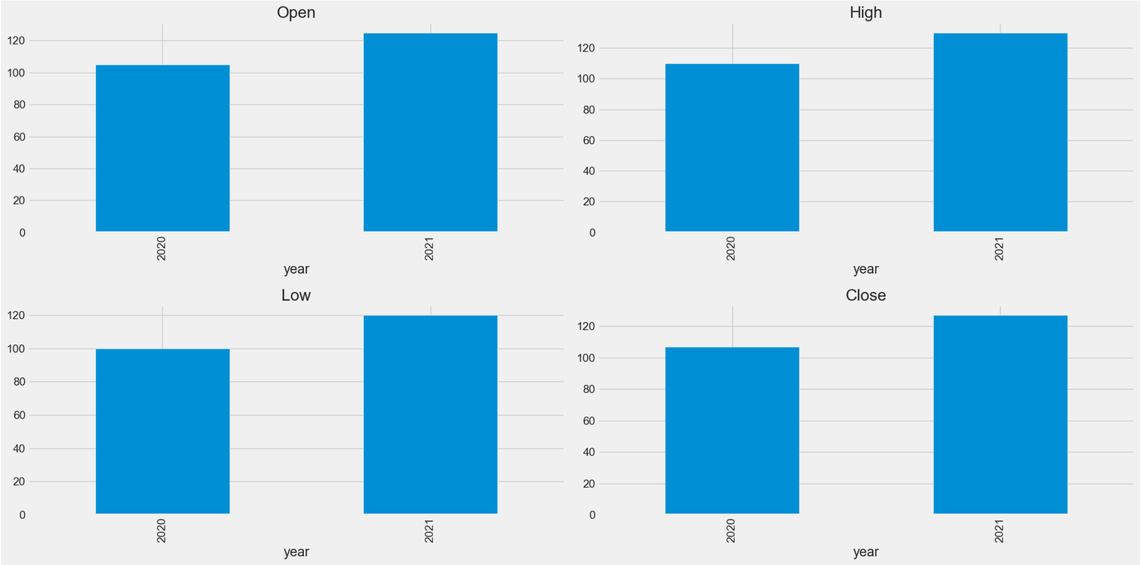


In [126]:
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sb.boxplot(df[col])
plt.show()
```



In [128]:

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

# Sample DataFrame
data = {'Date': ['2020-01-01', '2020-02-01', '2021-01-01', '2021-02-01'],
        'Open': [100, 110, 120, 130],
        'High': [105, 115, 125, 135],
        'Low': [95, 105, 115, 125],
        'Close': [102, 112, 122, 132]}

df = pd.DataFrame(data)

# Extract the year from the 'Date' column
df['year'] = pd.to_datetime(df['Date']).dt.year

# Now you can group by 'year'
data_grouped = df.groupby('year').mean()

# Create four subplots, one for each column
plt.figure(figsize=(20, 10))
for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
    plt.subplot(2, 2, i + 1)
    data_grouped[col].plot(kind='bar', title=col)

plt.tight_layout()
plt.show()
```



In [73]:

```
import pandas as pd

# Sample DataFrame
data = {'Date': ['01/15/2022', '02/20/2022', '03/2022', '04/05/2023']}
df = pd.DataFrame(data)

# Split the 'Date' column and handle NaN values
splitted = df['Date'].str.split('/', expand=True)
df['day'] = splitted[0].fillna(0).astype(int)
df['month'] = splitted[1].fillna(0).astype(int)
df['year'] = splitted[2].fillna(0).astype(int)

df.head()
```

Out[73]:

	Date	day	month	year
0	01/15/2022	1	15	2022
1	02/20/2022	2	20	2022
2	03/2022	3	2022	0
3	04/05/2023	4	5	2023

In [74]:

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

Out[74]:

	Date	day	month	year	is_quarter_end
0	01/15/2022	1	15	2022	1
1	02/20/2022	2	20	2022	0
2	03/2022	3	2022	0	1
3	04/05/2023	4	5	2023	0

In [87]:

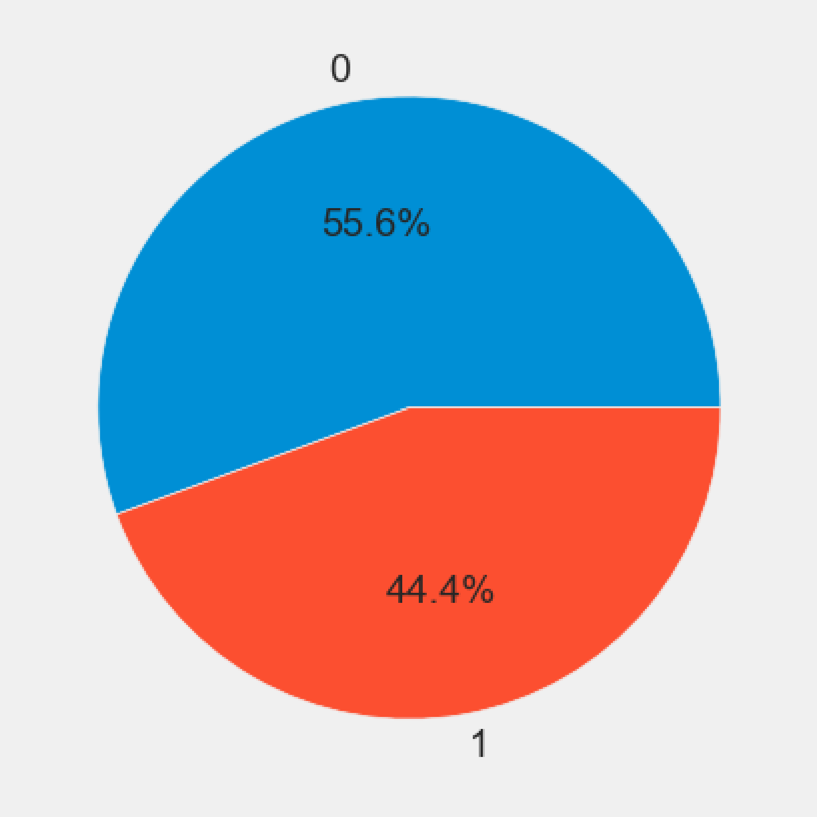
```
df.groupby('is_quarter_end').mean()
```

Out[87]:

	day	month	year
is_quarter_end			
0	3.0	12.5	2022.5
1	2.0	1018.5	1011.0

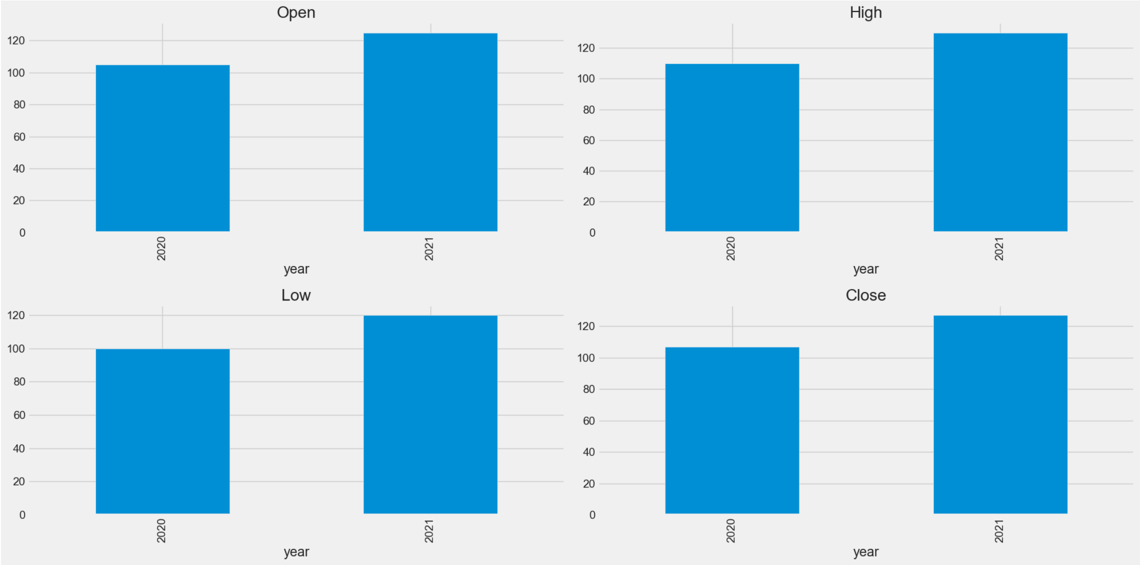
In [96]:

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

# Sample DataFrame
data = {'target': [0, 1, 0, 1, 0, 1, 0, 0, 1]}
df = pd.DataFrame(data)

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



In [100]:

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

# Sample DataFrame
data = {'Date': ['2020-01-01', '2020-02-01', '2021-01-01', '2021-02-01'],
        'Open': [100, 110, 120, 130],
        'High': [105, 115, 125, 135],
        'Low': [95, 105, 115, 125],
        'Close': [102, 112, 122, 132]}

df = pd.DataFrame(data)

# Extract the year from the 'Date' column
df['year'] = pd.to_datetime(df['Date']).dt.year

# Now you can group by 'year'
data_grouped = df.groupby('year').mean()

# Create four subplots, one for each column
plt.figure(figsize=(20, 10))
for i, col in enumerate(['Open', 'High', 'Low', 'Close']):
    plt.subplot(2, 2, i + 1)
    data_grouped[col].plot(kind='bar', title=col)

plt.tight_layout()
plt.show()
```

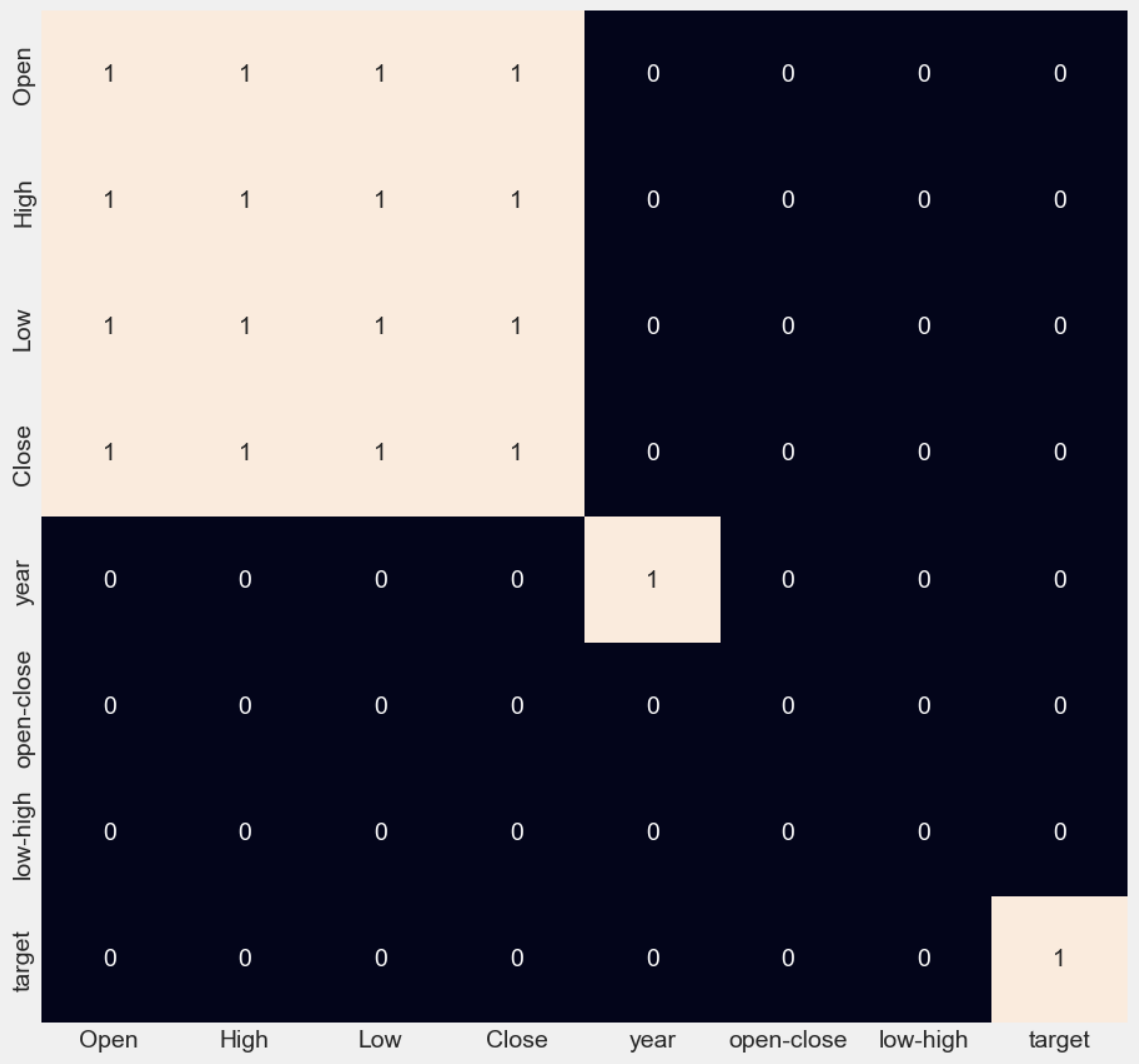


In [102]:

```
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 [103]:

```
plt.figure(figsize=(10, 10))

# As our concern is with the highly
# correlated features only so, we will visualize
# our heatmap as per that criteria only.
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
plt.show()
```



	Open	High	Low	Close	year	open-close	low-high	target
Open	1	1	1	1	0	0	0	0
High	1	1	1	1	0	0	0	0
Low	1	1	1	1	0	0	0	0
Close	1	1	1	1	0	0	0	0
year	0	0	0	0	1	0	0	0
open-close	0	0	0	0	0	0	0	0
low-high	0	0	0	0	0	0	0	0
target	0	0	0	0	0	0	0	1

In [107]:

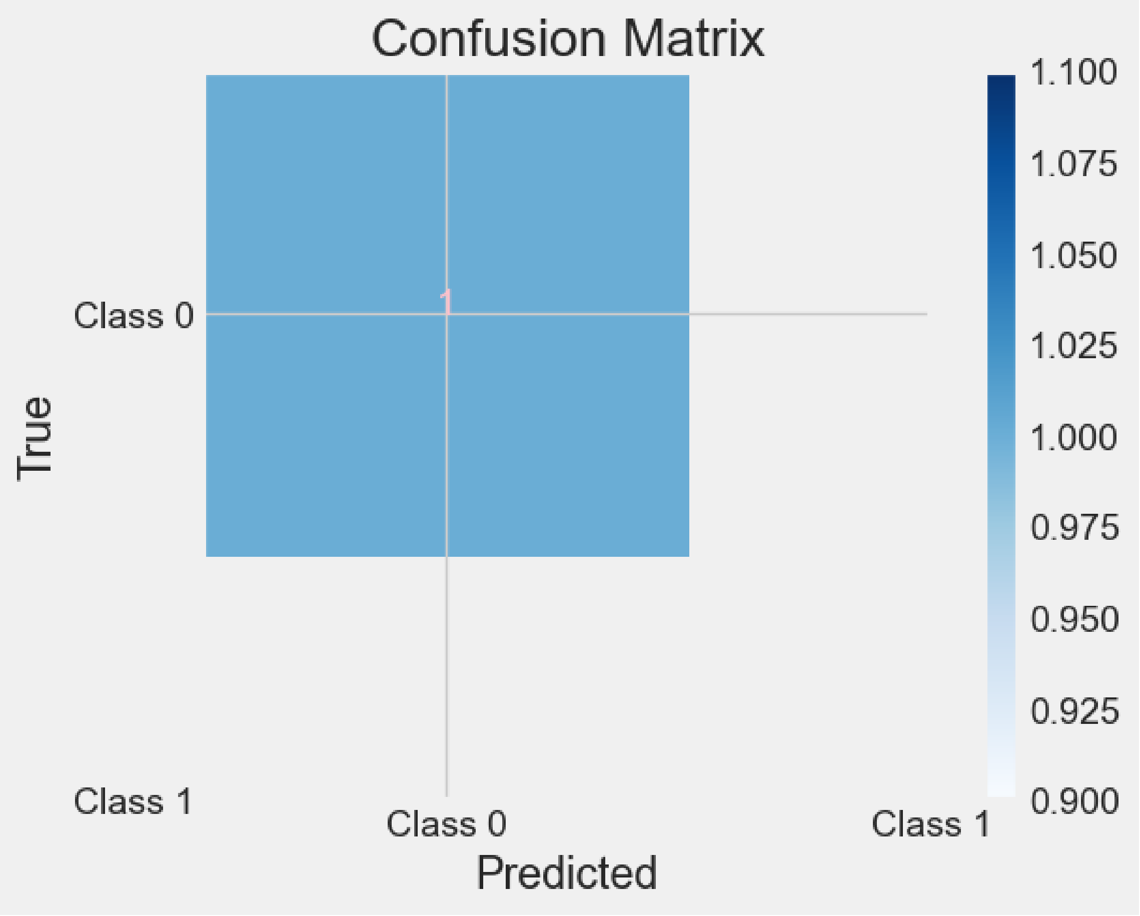
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Assuming you have a DataFrame 'df' with columns 'open-close', 'low-high',
# You may want to confirm the exact column names and their presence in your

# Select the features and target
features = df[['open-close', 'low-high']]  # You need to confirm that 'is_q
target = df['target']

# Scale the features using StandardScaler
scaler = StandardScaler()
features = scaler.fit_transform(features)

# Split the data into training and validation sets
X_train, X_valid, Y_train, Y_valid = train_test_split(
    features, target, test_size=0.1, random_state=2022)

# Print the shapes of the training and validation sets
print(X_train.shape, X_valid.shape)
```

```
(3, 2) (1, 2)
```

In [138]:

```
# Extract the year from the 'Date' column
df['year'] = pd.to_datetime(df['Date']).dt.year

# Now you can group by 'year'
data_grouped = df.groupby('year').mean()
```

```
In [147]: from sklearn.metrics import confusion_matrix
          import matplotlib.pyplot as plt
          import itertools

          # Assuming you have already trained the models and selected the first model
          # models[0] should be a classifier, e.g., Logistic Regression, SVC, or XGBo

          # Generate a confusion matrix for the first model on the validation dataset
          cm = confusion_matrix(Y_valid, models[0].predict(X_valid))

          # Create a confusion matrix plot
          plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
          plt.title("Confusion Matrix")
          plt.colorbar()

          # Set axis labels
          classes = ["Class 0", "Class 1"]
          tick_marks = range(len(classes))
          plt.xticks(tick_marks, classes)
          plt.yticks(tick_marks, classes)

          # Display the values in the cells
          thresh = cm.max() / 2
          for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
              plt.text(j, i, cm[i, j], horizontalalignment="center", color="pink" if

          plt.xlabel("Predicted")
          plt.ylabel("True")
          plt.show()
```



In [ ]:

```

```