In [48]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
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


df =pd.read_csv (r'C:\Users\victo\Downloads\Tesla.csv')


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

In [49]:
```
df
```

Out[49]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.000000	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.420000	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.920000	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.100000	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.000000	15.830000	16.110001	6866900	16.110001
...	...	...	...	...	...	...	...
1687	3/13/2017	244.820007	246.850006	242.779999	246.169998	3010700	246.169998
1688	3/14/2017	246.110001	258.119995	246.020004	258.000000	7575500	258.000000
1689	3/15/2017	257.000000	261.000000	254.270004	255.729996	4816600	255.729996
1690	3/16/2017	262.399994	265.750000	259.059998	262.049988	7100400	262.049988
1691	3/17/2017	264.000000	265.329987	261.200012	261.500000	6475900	261.500000

1692 rows × 7 columns

In [50]:
```
df.head()
```

Out[50]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001

In [51]:
```
df.shape
```

Out[51]:
```
(1692, 7)
```

In [52]:
```
df.describe()
```

Out[52]:

	Open	High	Low	Close	Volume	Adj Close
count	1692.000000	1692.000000	1692.000000	1692.000000	1.692000e+03	1692.000000
mean	132.441572	134.769698	129.996223	132.428658	4.270741e+06	132.428658
std	94.309923	95.694914	92.855227	94.313187	4.295971e+06	94.313187
min	16.139999	16.629999	14.980000	15.800000	1.185000e+05	15.800000
25%	30.000000	30.650000	29.215000	29.884999	1.194350e+06	29.884999
50%	156.334999	162.370002	153.150002	158.160004	3.180700e+06	158.160004
75%	220.557495	224.099999	217.119999	220.022503	5.662100e+06	220.022503
max	287.670013	291.420013	280.399994	286.040009	3.716390e+07	286.040009

In [53]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1692 entries, 0 to 1691
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1692 non-null   object 
 1   Open       1692 non-null   float64
 2   High       1692 non-null   float64
 3   Low        1692 non-null   float64
 4   Close      1692 non-null   float64
 5   Volume     1692 non-null   int64  
 6   Adj Close  1692 non-null   float64
dtypes: float64(5), int64(1), object(1)
memory usage: 92.7+ KB
```

Exploratory Data Analysis, while performing the EDA of the Tesla stock price, we will nalyzw how prices of the stock have moved

over the period of time and how the ends of the quarters affects the prices of the stock

In [54]:
```
plt.figure(figsize=(15,5))
plt.plot(df['Close'])
plt.title('Tesla Close Price',fontsize=15)
plt.ylabel('Price in dollars')
plt.show()
```

In [55]:
```
df[df['Close'] == df['Adj Close']].shape
```

Out[55]:
```
(1692, 7)
```

In [56]:
```
df.drop(['Adj Close'],axis=1)
```

Out[56]:

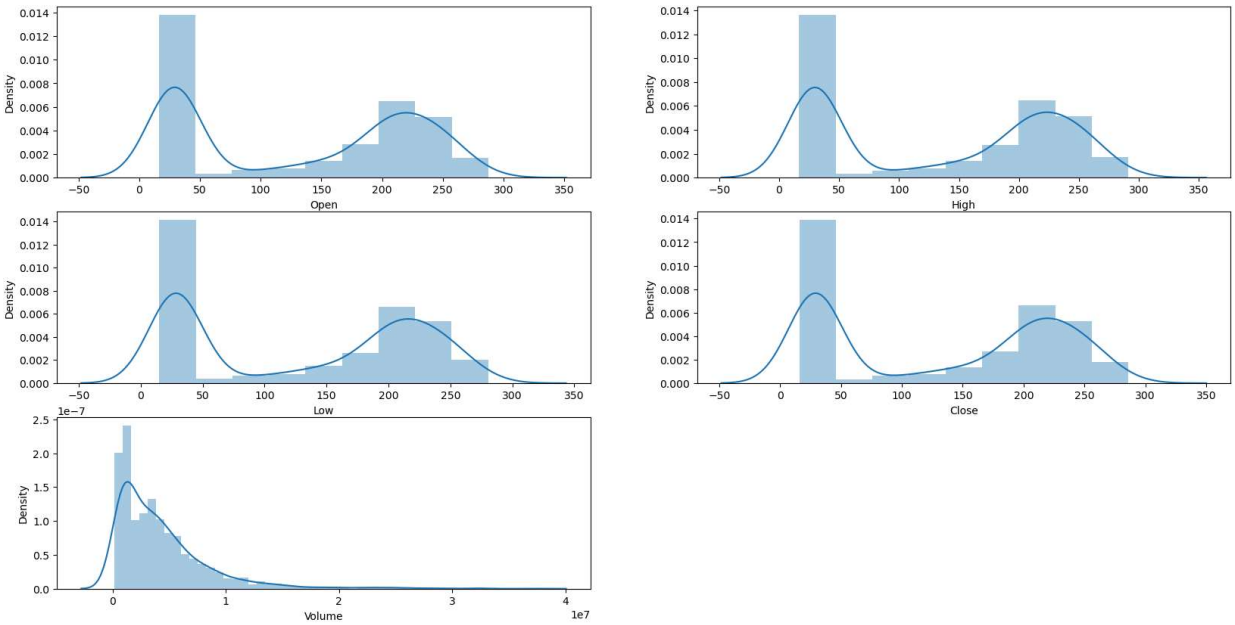
	Date	Open	High	Low	Close	Volume
0	6/29/2010	19.000000	25.000000	17.540001	23.889999	18766300
1	6/30/2010	25.790001	30.420000	23.299999	23.830000	17187100
2	7/1/2010	25.000000	25.920000	20.270000	21.959999	8218800
3	7/2/2010	23.000000	23.100000	18.709999	19.200001	5139800
4	7/6/2010	20.000000	20.000000	15.830000	16.110001	6866900
...	...	...	...	...	...	...
1687	3/13/2017	244.820007	246.850006	242.779999	246.169998	3010700
1688	3/14/2017	246.110001	258.119995	246.020004	258.000000	7575500
1689	3/15/2017	257.000000	261.000000	254.270004	255.729996	4816600
1690	3/16/2017	262.399994	265.750000	259.059998	262.049988	7100400
1691	3/17/2017	264.000000	265.329987	261.200012	261.500000	6475900

1692 rows × 6 columns

In [57]:
```
df.isnull().sum()
```

Out[57]:
```
Date         0
Open         0
High         0
Low          0
Close        0
Volume       0
Adj Close    0
dtype: int64
```

In [58]:
```
features = ['Open','High','Low','Close','Volume']
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(3,2,i+1)
    sb.distplot(df[col])
plt.show()
```



In [59]:
```
splitter = df['Date'].str.split('/',expand=True)
df['day'] = splitter[1].astype('int')
df['month'] = splitter[0].astype('int')
df['year'] = splitter[2].astype('int')
```

In [60]:
```
df.head()
```

Out[60]:

	Date	Open	High	Low	Close	Volume	Adj Close	day	month	year
0	6/29/2010	19.000000	25.00	17.540001	23.889999	18766300	23.889999	29	6	2010
1	6/30/2010	25.790001	30.42	23.299999	23.830000	17187100	23.830000	30	6	2010
2	7/1/2010	25.000000	25.92	20.270000	21.959999	8218800	21.959999	1	7	2010
3	7/2/2010	23.000000	23.10	18.709999	19.200001	5139800	19.200001	2	7	2010
4	7/6/2010	20.000000	20.00	15.830000	16.110001	6866900	16.110001	6	7	2010

In [61]:
```
df['is_quarter_end'] = np.where(df['month']%3==0,1,0)
```

df.head()

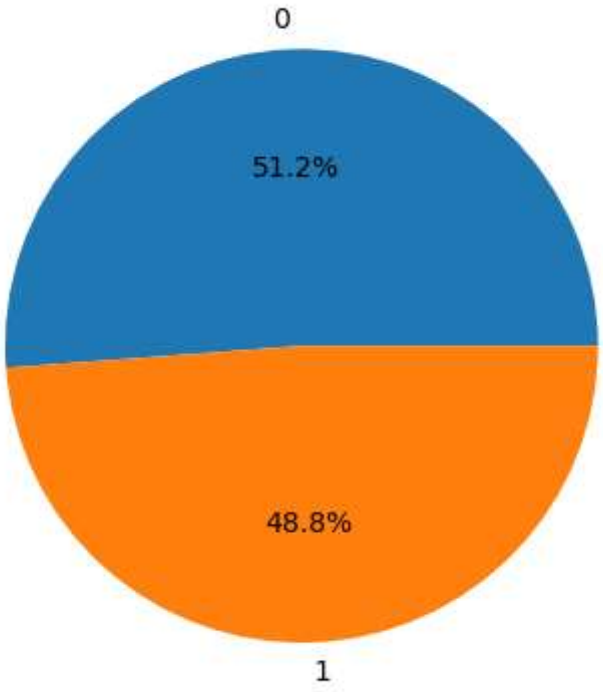
In [62]:
```
df.drop('Date',axis=1).groupby('is_quarter_end').mean()
```

Out[62]:

is_quarter_end	Open	High	Low	Close	Volume	Adj Close	day
0	130.813739	133.182620	128.257229	130.797709	4.461581e+06	130.797709	15.686501
1	135.679982	137.927032	133.455777	135.673269	3.891084e+06	135.673269	15.657244

In [63]:
```
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 [105…
```
plt.pie(df['target'].value_counts().values,labels=[0,1],autopct='%1.1f%%')
plt.show()
```



In [124…
```
features = df[['open-close','low-high','is_quarter_end']]
target = df['target']
```

In [141…
```
pipe = Pipeline([
    ('preprocessing',StandardScaler()),
    ('classifier',XGBClassifier())
])
```

In [147…
```
#Note that each classifier must be put in a dictionary
param_grid = [
```

```
    # For XGBClassifier
    {
        'preprocessing': [StandardScaler(), None],
        'classifier': [XGBClassifier(eval_metric='logloss')],
        'classifier__max_depth': [3, 5, 7],
        'classifier__learning_rate': [0.01, 0.1]
    },
    # For LogisticRegression
    {
        'preprocessing': [StandardScaler(), None],
        'classifier': [LogisticRegression(max_iter=1000)],
        'classifier__C': [0.1, 1, 10],
        'classifier__penalty': ['l2']
    },
    # For SVC
    {
        'preprocessing': [StandardScaler(),None],
        'classifier': [SVC(probability=True)],
        'classifier__kernel': [ 'poly'],

    }
]
```

In [148…
```
X_train,X_test,y_train,y_test = train_test_split(features,target,random_state=2022,tes
```

In [149…
```
grid = GridSearchCV(pipe, param_grid, scoring='roc_auc', n_jobs=-1)
```

In [150…
```
vics = grid.fit(X_train,y_train)
```

In [151…
```
print("Best params:\n{}\n".format(grid.best_params_))
```

```
Best params:
{'classifier': LogisticRegression(C=0.1, max_iter=1000), 'classifier__C': 0.1, 'class
ifier__penalty': 'l2', 'preprocessing': None}
```

In [134…
```
print("Best cross-validation score: {:.2f}".format(grid.best_score_))
```

```
Best cross-validation score: 0.51
```

In [135…
```
print('Test-set score:{:.2f}'.format(grid.score(X_test,y_test)))
```

```
Test-set score:0.55
```

In [102…
```
print('Training score:{:.2f}'.format(grid.score(X_train,y_train)))
```

```
Training score:0.52
```

In [92]:
```
y_pred = vics.predict(X_test)
```

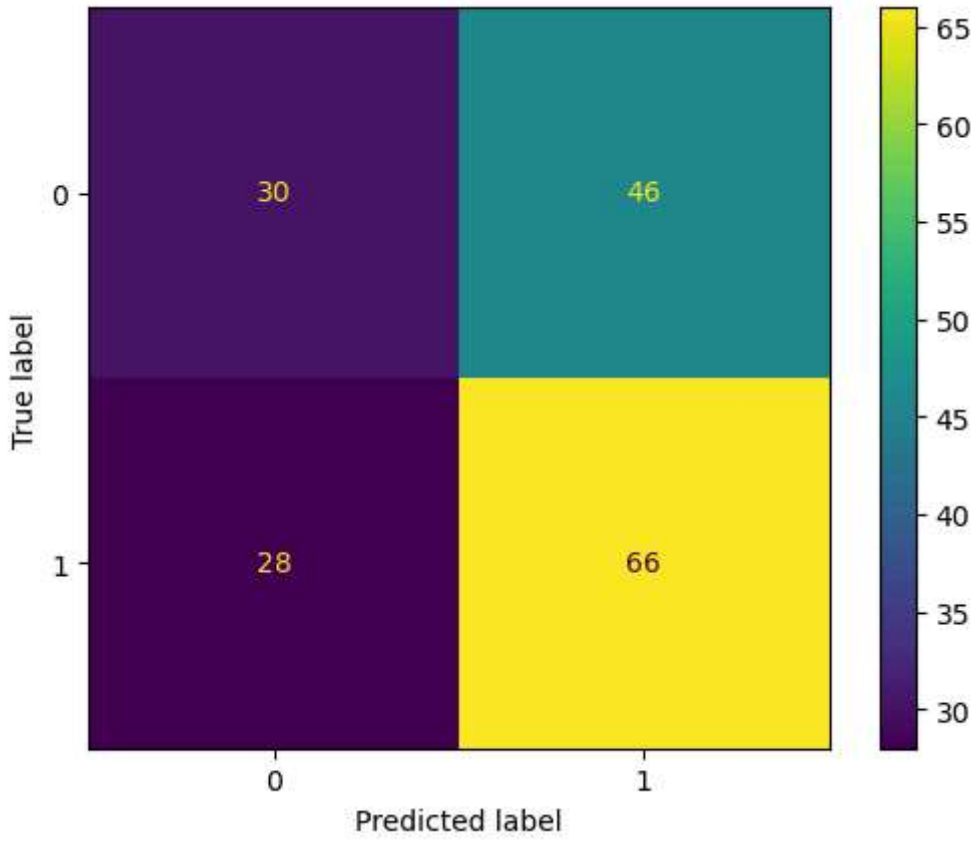
In [ ]:
```

```

In [136…
```
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(y_test,y_pred)
print("Confusion matrix:\n{}".format(confusion))
```

```
Confusion matrix:
[[30 46]
 [28 66]]
```

In [122…
```
from sklearn.metrics import ConfusionMatrixDisplay
display = ConfusionMatrixDisplay(confusion_matrix=confusion)
display.plot()
plt.show()
```



In [123…
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.52      0.39      0.45        76
           1       0.59      0.70      0.64        94

    accuracy                           0.56       170
   macro avg       0.55      0.55      0.54       170
weighted avg       0.56      0.56      0.55       170
```

In [45]:
```
from sklearn.metrics import precision_score

print("Precision (micro):", precision_score(y_test, victor, average='micro'))
print("Precision (macro):", precision_score(y_test, victor, average='macro'))
print("Precision (weighted):", precision_score(y_test, victor, average='weighted'))
```

```
Precision (micro): 0.5941176470588235
Precision (macro): 0.5899315738025415
Precision (weighted): 0.5946409062158589
```

In [ ]: