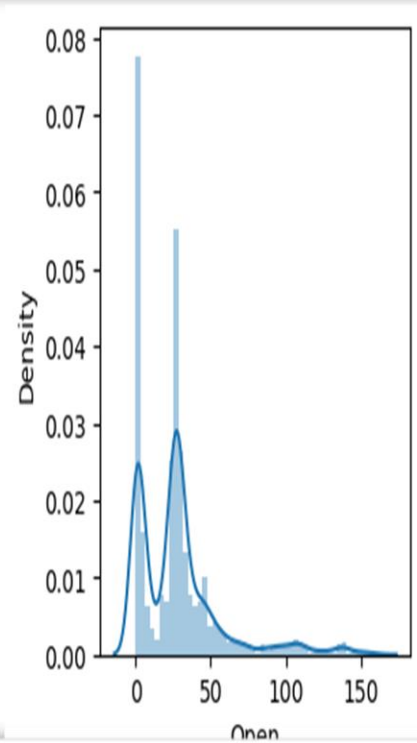
PHASE-5
STOCK
PRICE PREDICTION

In [21]:
```
import seaborn as sb
```

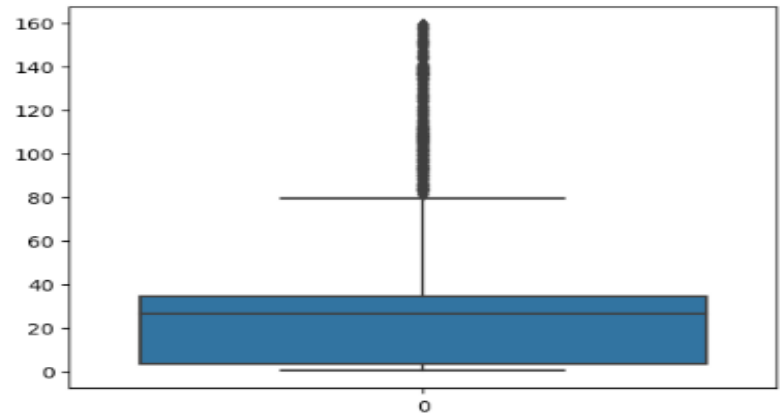
In [21]:
```
import seaborn as sb
```

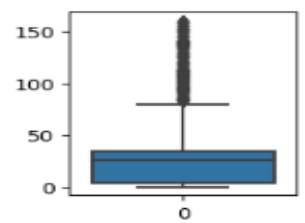
In [31]:
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']
plt.subplots(figsize=(10,8))
for i, col in enumerate(features):
    plt.subplot(2,3,i+1)
    sb.distplot(stock[col])
    plt.show()
```

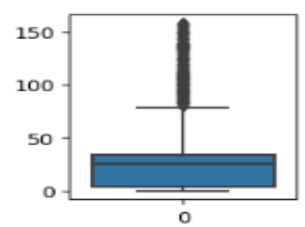


In [24]:
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2,3,i+1)
    sb.boxplot(stock[col])
    plt.show()
```

```
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\1085974214.py:3: MatplotlibDeprecationWarning: Auto-removal of overlapping a
xes 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)
```

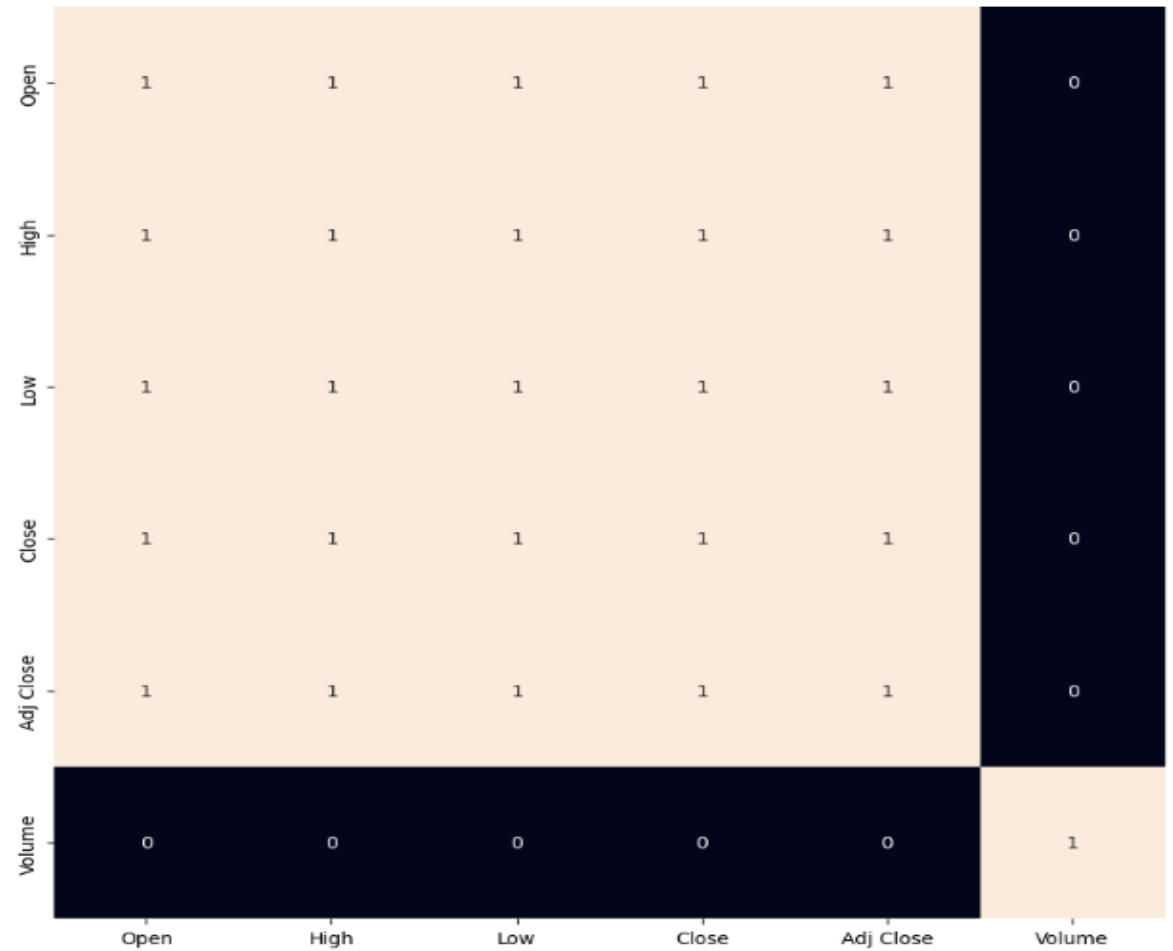






In [26]:
```
plt.figure(figsize=(10, 10))
sb.heatmap(stock.corr() > 0.9, annot=True, cbar=False)
plt.show()
```

```
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\4185554148.py:2: FutureWarning: The default value of numeric_only in DataFra
me.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric
_only to silence this warning.
  sb.heatmap(stock.corr() > 0.9, annot=True, cbar=False)
```



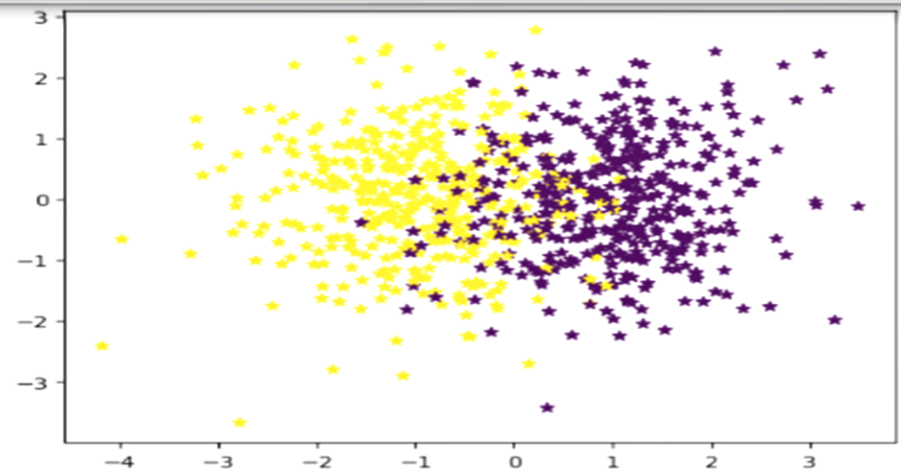
In [28]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import numpy as np
np.random.seed(0)
data = {
    'Exam1': np.random.rand(100) * 100,
    'Exam2': np.random.rand(100) * 100,
    'Admitted': np.random.randint(2, size=100)
}
df = pd.DataFrame(data)
print(df)
X = df[['Exam1', 'Exam2']]
y = df['Admitted']
print(X)
print(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("------------------")
print(y_pred)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
        Exam1      Exam2  Admitted
0   54.881350  67.781654         0
1   71.518937  27.000797         1
2   60.276338  73.519402         0
3   54.488318  96.218855         0
4   42.365480  24.875314         1
..        ...        ...       ...
95  18.319136  49.045881         0
96  58.651293  22.741463         1
97   2.010755  25.435648         0
98  82.894003   5.802916         1
99   0.469548  43.441663         0

[100 rows x 3 columns]
        Exam1      Exam2
0   54.881350  67.781654
1   71.518937  27.000797
2   60.276338  73.519402
3   54.488318  96.218855
4   42.365480  24.875314
..        ...        ...
95  18.319136  49.045881
96  58.651293  22.741463
97   2.010755  25.435648
98  82.894003   5.802916
99   0.469548  43.441663

[100 rows x 2 columns]
0     0
1     1
2     0
3     0
4     1
     ..
95    0
96    1
97    0
98    1
99    0
Name: Admitted, Length: 100, dtype: int32
------------------
[1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1]
Accuracy: 0.45
              precision    recall  f1-score   support

           0       0.27      0.50      0.35         6
           1       0.67      0.43      0.52        14

    accuracy                           0.45        20
   macro avg       0.47      0.46      0.44        20
weighted avg       0.55      0.45      0.47        20

[[3 3]
 [8 6]]
```

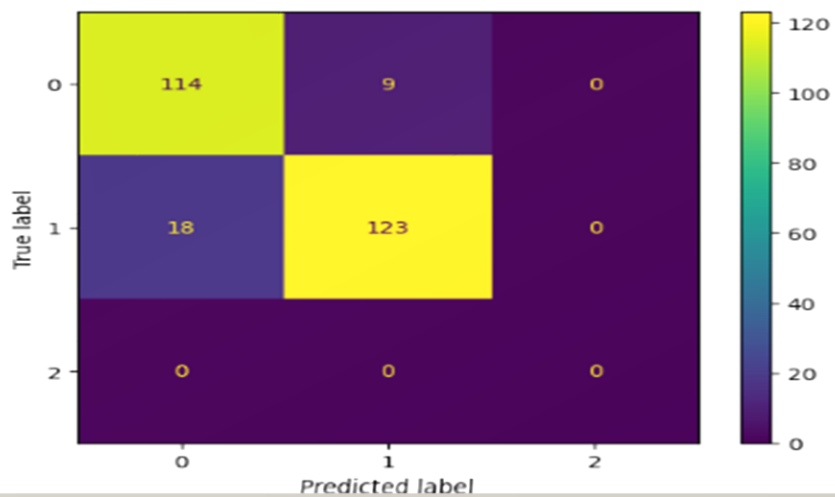
In [29]:

```
from sklearn.datasets import make_classification
value1, y = make_classification(n_features=6,n_classes=2,n_samples=800,n_informative=2,random_state=66,n_clusters_per_class=1)
import matplotlib.pyplot as plt
plt.scatter(value1[:, 0], value1[:, 1], c=y, marker="*")
plt.show()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(value1, y, test_size=0.33, random_state=125)
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X_train, y_train)
predicted = model.predict([X_test[6]])
print("Actual Value:", y_test[6])
print("Predicted Value:", predicted[0])
from sklearn.metrics import (accuracy_score,confusion_matrix,ConfusionMatrixDisplay,f1_score,)
y_pred = model.predict(X_test)
accuray = accuracy_score(y_pred, y_test)
f1 = f1_score(y_pred, y_test, average="weighted")
print("Accuracy:", accuray)
print("F1 Score:", f1)
labels = [0,1,2]
cm = confusion_matrix(y_test, y_pred, labels=labels)
print(cm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot()
```



```
Actual Value: 1
Predicted Value: 1
Accuracy: 0.8977272727272727
F1 Score: 0.8976082740788625
[[114   9   0]
 [ 18 123   0]
 [  0   0   0]]
```

Out[29]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x26d022e7d50>



CONCLUSION

- To preprocess the dataset for stock price prediction, you can consider steps like removing duplicates, handling missing values, scaling the data, and splitting it into training and testing sets.

Team Member

Leader: Shaik.Nawab Sohail

Member:Shaik.Imran

Member: Mohammed Madhar

Member: Mohammed Abdul Raheem