```
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')

from google.colab import files
uploaded = files.upload()

import io
df = pd.read_csv(io.BytesIO(uploaded['google.csv']))
print(df.head())
print('\n', df)
from sklearn.preprocessing import MinMaxScaler
from keras.layers import LSTM, Dense, Dropout
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib. dates as mandates
from sklearn.preprocessing import MinMaxScaler
from sklearn import linear_model
from keras.models import Sequential
from keras.layers import Dense
import keras.backend as K
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from keras.models import load_model
from keras.layers import LSTM
from keras.utils.vis_utils import plot_model
```

Choose Files google.csv

- **google.csv**(text/csv) - 158056 bytes, last modified: 11/10/2022 - 100% done

```
Saving google.csv to google.csv
        Date    Open    High     Low   Close   Volume  Year
0  30-Jun-17  943.99  945.00  929.61  929.68  2287662  2017
1  29-Jun-17  951.35  951.66  929.60  937.82  3206674  2017
2  28-Jun-17  950.66  963.24  936.16  961.01  2745568  2017
3  27-Jun-17  961.60  967.22  947.09  948.09  2443602  2017
```

```
df.shape
```

```
(3145, 7)
```

```
1     29-Jun-17  951.35  951.66  929.60  937.82   3206674  2017
```

```
df.describe()
```

	Open	High	Low	Close	Volume	Year
count	3145.000000	3145.000000	3145.000000	3145.000000	3.145000e+03	3145.000000
mean	382.514169	385.872099	378.737126	382.350248	4.205708e+06	2010.759300
std	213.520466	214.636421	212.113835	213.469899	3.878100e+06	3.614485
min	87.740000	89.290000	86.370000	87.580000	5.211410e+05	2005.000000
25%	232.380000	234.890000	230.400000	232.440000	1.889613e+06	2008.000000
50%	296.280000	298.520000	293.640000	296.050000	2.811069e+06	2011.000000
75%	544.000000	548.220000	539.850000	543.650000	5.232088e+06	2014.000000
max	1005.490000	1008.610000	996.620000	1004.280000	4.118289e+07	2017.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3145 entries, 0 to 3144
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    3145 non-null   object
 1   Open    3145 non-null   float64
 2   High    3145 non-null   float64
 3   Low     3145 non-null   float64
 4   Close   3145 non-null   float64
 5   Volume  3145 non-null   int64
 6   Year    3145 non-null   int64
dtypes: float64(4), int64(2), object(1)
memory usage: 172.1+ KB
```

```
plt.figure(figsize=(20,10))
plt.plot(df['Close'])
plt.title('Google Close price.', fontsize=18)
plt.ylabel('Price in dollars.')
plt.show()
```

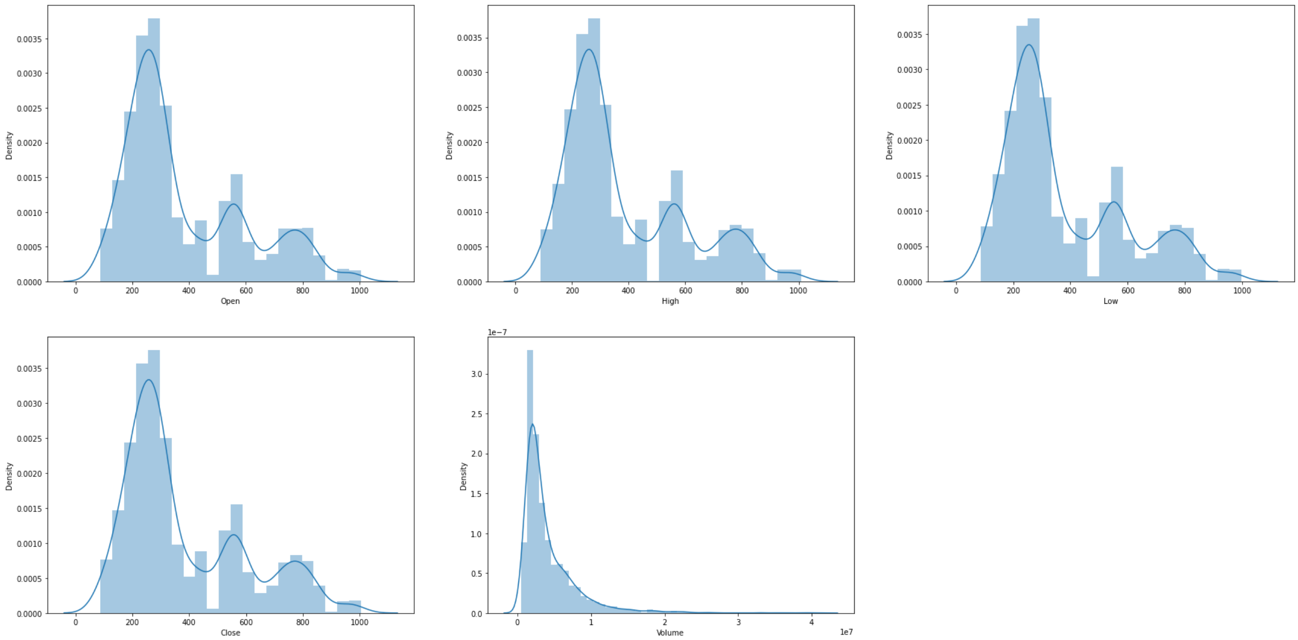

```
df.head()
```

	Date	Open	High	Low	Close	Volume	Year
0	30-Jun-17	943.99	945.00	929.61	929.68	2287662	2017
1	29-Jun-17	951.35	951.66	929.60	937.82	3206674	2017
2	28-Jun-17	950.66	963.24	936.16	961.01	2745568	2017
3	27-Jun-17	961.60	967.22	947.09	948.09	2443602	2017
4	26-Jun-17	990.00	993.99	970.33	972.09	1517912	2017

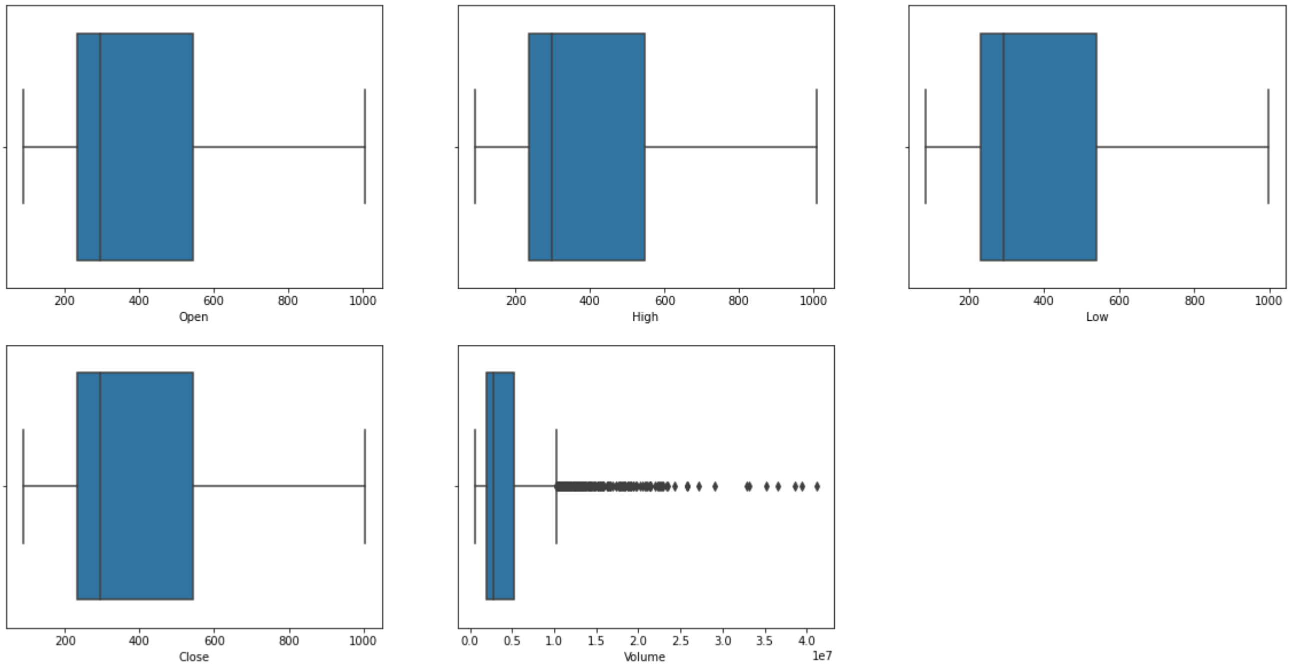
```
df.isnull().sum()
```

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

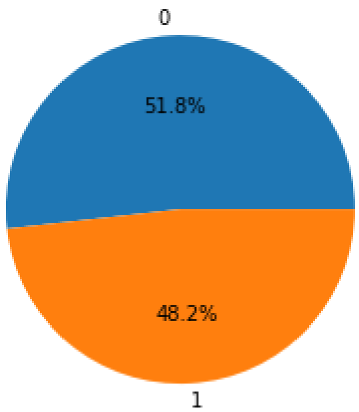
```
features = ['Open', 'High', 'Low', 'Close', 'Volume']

plt.subplots(figsize=(30,15))

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




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

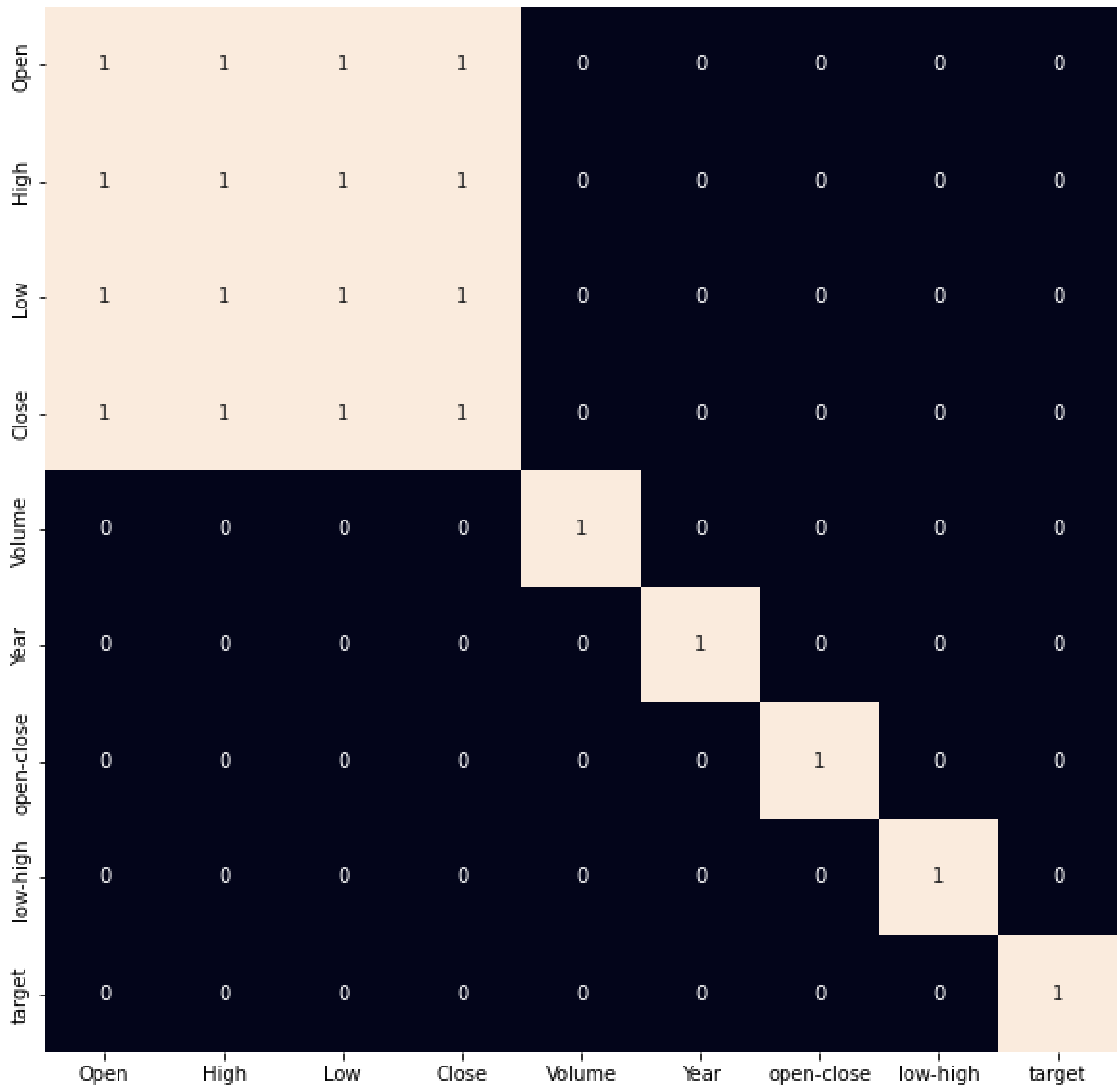



```
df['open-close']  = df['Open'] - df['Close']
df['low-high']  = df['Low'] - df['High']
df['target'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)
plt.pie(df['target'].value_counts().values,
        labels=[0, 1], autopct='%1.1f%%')
plt.show()
```



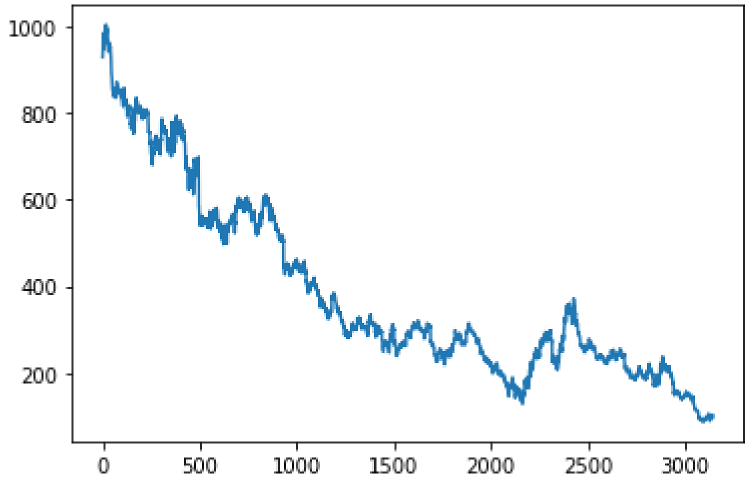

```
plt.figure(figsize=(10, 10))
```

```
sb.heatmap(df.corr() > 0.9, annot=True, cbar=False)
plt.show()
```




```
df['Close'].plot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fabef61b450>




```
#Scaling
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
feature_transform = scaler.fit_transform(df[features])
feature_transform= pd.DataFrame(columns=features, data=feature_transform, index=df.index)
feature_transform.head()
```

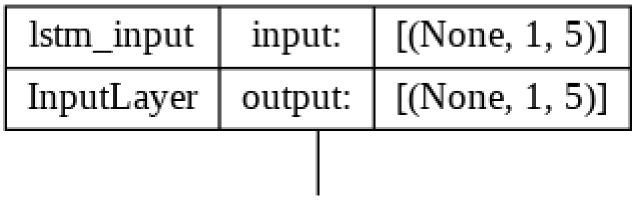
	Open	High	Low	Close	Volume
0	0.932988	0.930808	0.926383	0.918621	0.043444
1	0.941008	0.938052	0.926372	0.927501	0.066046
2	0.940256	0.950648	0.933579	0.952798	0.054706
3	0.952177	0.954978	0.945586	0.938704	0.047279
4	0.983122	0.984097	0.971118	0.964885	0.024514

```
#Set Target Variable
output_var = pd.DataFrame(df['Close'])
#Selecting the Features
features = ['Open', 'High', 'Low', 'Volume']
```

```
#Splitting to Training set and Test set
timesplit= TimeSeriesSplit(n_splits=10)
for train_index, test_index in timesplit.split(feature_transform):
        X_train, X_test = feature_transform[:len(train_index)], feature_transform[len(trai
        y_train, y_test = output_var[:len(train_index)].values.ravel(), output_var[len(tra
```

```
#Process the data for LSTM
trainX =np.array(X_train)
testX =np.array(X_test)
X_train = trainX.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test = testX.reshape(X_test.shape[0], 1, X_test.shape[1])
```

```
#Building the LSTM Model
lstm = Sequential()
lstm.add(LSTM(32, input_shape=(1, trainX.shape[1]), activation='relu', return_sequences=Fa
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error', optimizer='adam')
plot_model(lstm, show_shapes=True, show_layer_names=True)
```




```
history=lstm.fit(X_train, y_train, epochs=100, batch_size=8, verbose=1, shuffle=False)
```

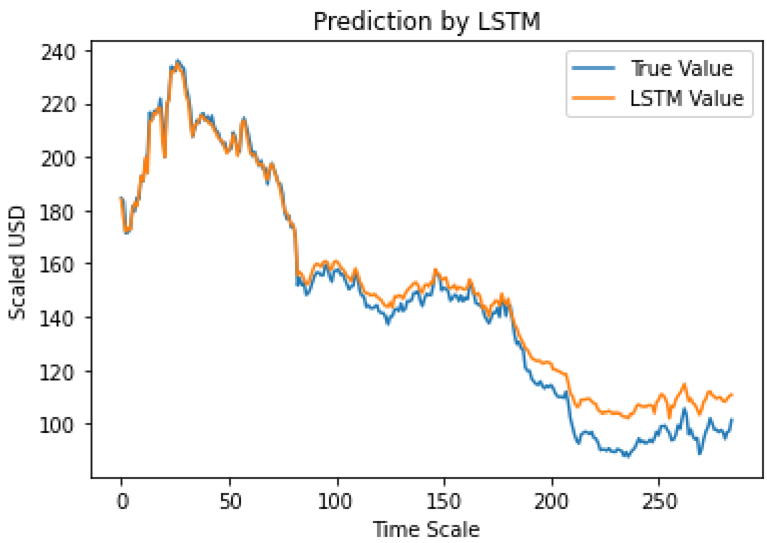
```
Epoch 72/100
358/358 [==============================] - 1s 3ms/step - loss: 15.1201
Epoch 73/100
358/358 [==============================] - 1s 3ms/step - loss: 15.0143
Epoch 74/100
358/358 [==============================] - 1s 3ms/step - loss: 14.9083
Epoch 75/100
358/358 [==============================] - 1s 3ms/step - loss: 14.8014
Epoch 76/100
358/358 [==============================] - 1s 3ms/step - loss: 14.6936
Epoch 77/100
358/358 [==============================] - 1s 3ms/step - loss: 14.5846
Epoch 78/100
358/358 [==============================] - 1s 3ms/step - loss: 14.4744
Epoch 79/100
358/358 [==============================] - 1s 3ms/step - loss: 14.3626
Epoch 80/100
358/358 [==============================] - 1s 3ms/step - loss: 14.2493
Epoch 81/100
358/358 [==============================] - 1s 3ms/step - loss: 14.1345
Epoch 82/100
358/358 [==============================] - 1s 3ms/step - loss: 14.0179
Epoch 83/100
358/358 [==============================] - 1s 3ms/step - loss: 13.8997
Epoch 84/100
358/358 [==============================] - 1s 3ms/step - loss: 13.7798
Epoch 85/100
358/358 [==============================] - 1s 3ms/step - loss: 13.6583
Epoch 86/100
358/358 [==============================] - 1s 3ms/step - loss: 13.5352
Epoch 87/100
358/358 [==============================] - 1s 3ms/step - loss: 13.4107
Epoch 88/100
358/358 [==============================] - 1s 3ms/step - loss: 13.2848
Epoch 89/100
358/358 [==============================] - 1s 3ms/step - loss: 13.1576
Epoch 90/100
358/358 [==============================] - 1s 3ms/step - loss: 13.0291
Epoch 91/100
358/358 [==============================] - 1s 3ms/step - loss: 12.8996
Epoch 92/100
358/358 [==============================] - 1s 3ms/step - loss: 12.7691
Epoch 93/100
358/358 [==============================] - 1s 3ms/step - loss: 12.6377
Epoch 94/100
358/358 [==============================] - 1s 3ms/step - loss: 12.5056
Epoch 95/100
358/358 [==============================] - 1s 3ms/step - loss: 12.3730
Epoch 96/100
358/358 [==============================] - 1s 3ms/step - loss: 12.2398
Epoch 97/100
358/358 [==============================] - 1s 3ms/step - loss: 12.1063
```

```
Epoch 98/100
358/358 [==============================] - 1s 3ms/step - loss: 11.9726
Epoch 99/100
358/358 [==============================] - 1s 3ms/step - loss: 11.8388
Epoch 100/100
358/358 [==============================] - 1s 3ms/step - loss: 11.7050
```

```
#LSTM Prediction
y_pred= lstm.predict(X_test)
```

```
9/9 [==============================] - 0s 2ms/step
```

```
#Predicted vs True Adj Close Value – LSTM
plt.plot(y_test, label='True Value')
plt.plot(y_pred, label='LSTM Value')
plt.title("Prediction by LSTM")
plt.xlabel('Time Scale')
plt.ylabel('Scaled USD')
plt.legend()
plt.show()
```



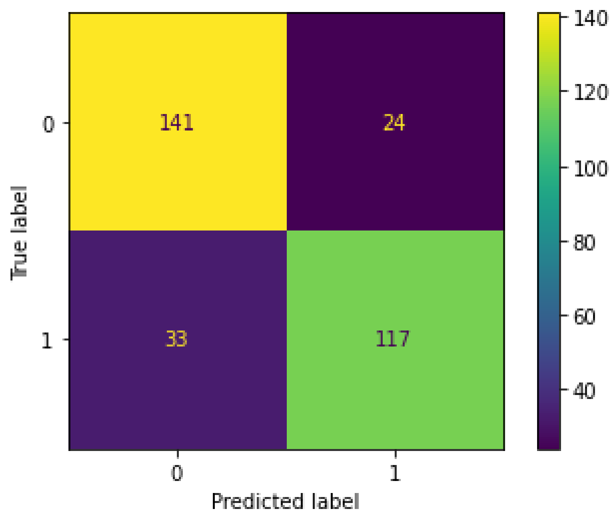

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn import metrics
```

```
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)

features = df[['open-close', 'low-high', 'Year']]
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,'\n\n')
models = [LogisticRegression(), SVC(
  kernel='poly', probability=True), XGBClassifier()]

for i in range(3):
  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()
```

```
(2830, 3) (315, 3)


LogisticRegression() :
Training Accuracy :  0.9115537167270719
Validation Accuracy :  0.9145858585858586

SVC(kernel='poly', probability=True) :
Training Accuracy :  0.8955785609133604
Validation Accuracy :  0.8862020202020202

XGBClassifier() :
Training Accuracy :  0.9366449247533783
Validation Accuracy :  0.9161212121212121
```

```
metrics.plot_confusion_matrix(models[0], X_valid, Y_valid)
plt.show()
```



0s completed at 9:14 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.