

# STOCK PRICE PREDICTION



Presented by :R.Nandhini

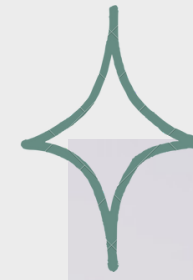
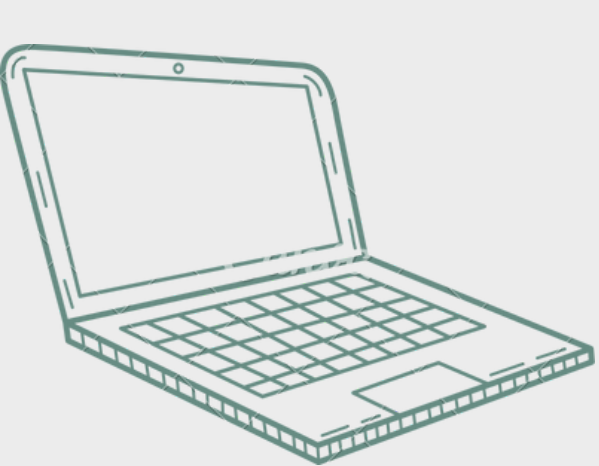
# INTRODUCTION:

**Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assetstraded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do text**



# ABSTRACT

- **In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values.**



# EXPLORATORY ANALYSIS

- **To begin this exploratory analysis, first import libraries and define functions for plotting the data using matplotlib. Depending on the data, not all plots will be made.**





# PROGRAM

```
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  
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()
```





# OUTPUT



```
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()
```





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

```
plt.subplots(figsize=(20,10))
```

```
for i, col in enumerate(features):
```

```
    plt.subplot(2,3,i+1)
```

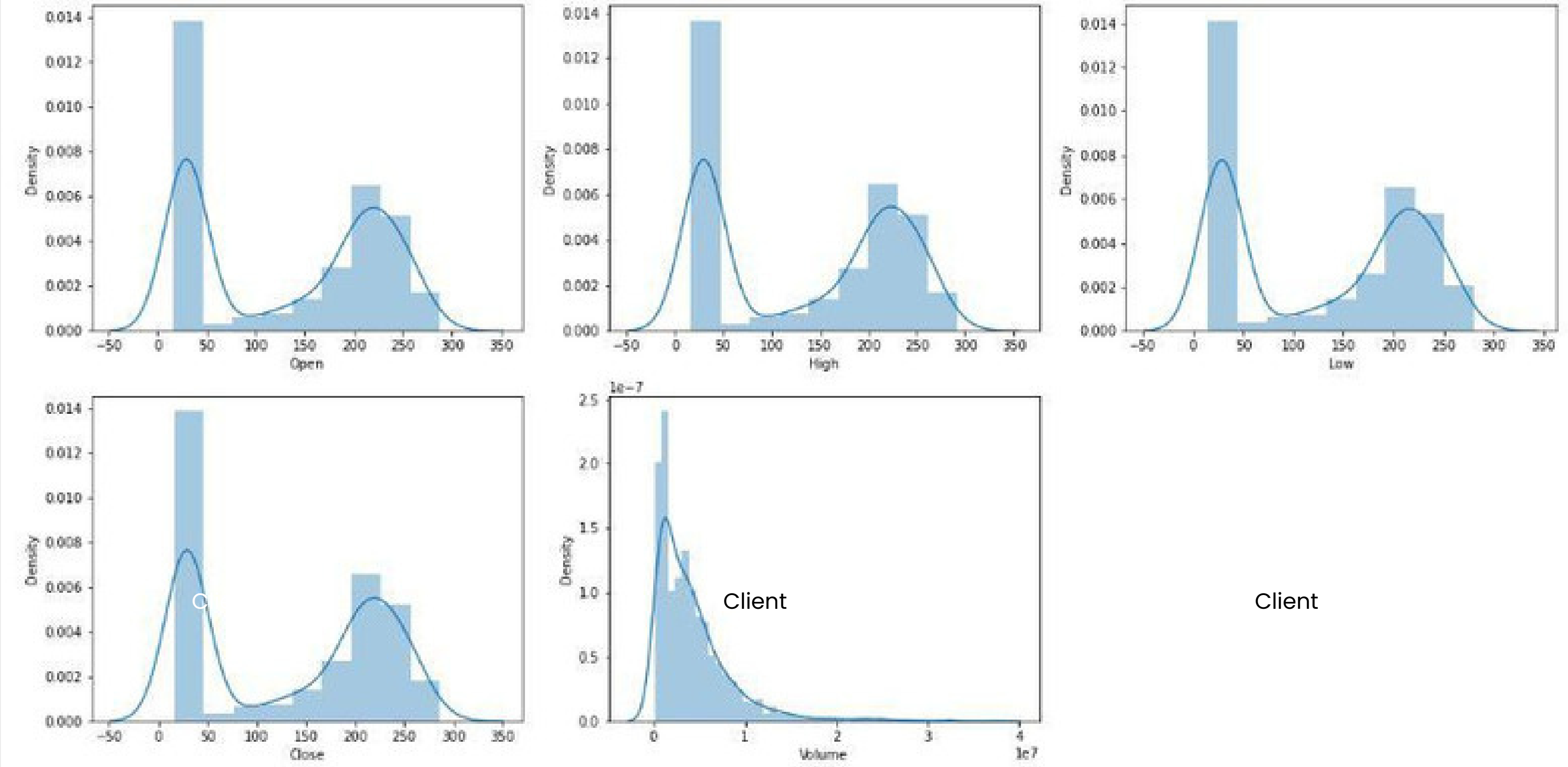
```
    sb.distplot(df[col])
```

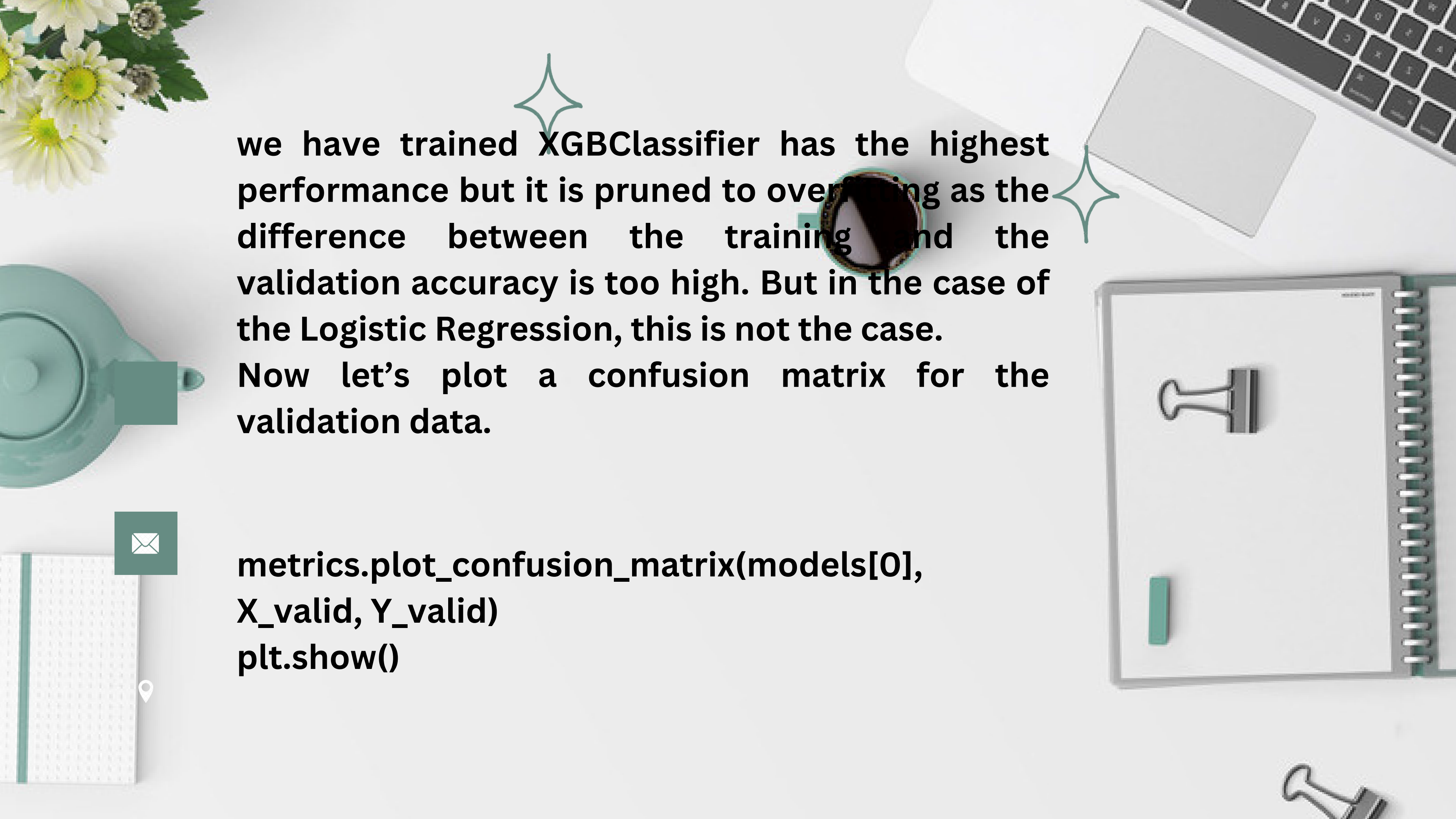
```
plt.show()
```





# OUTPUT

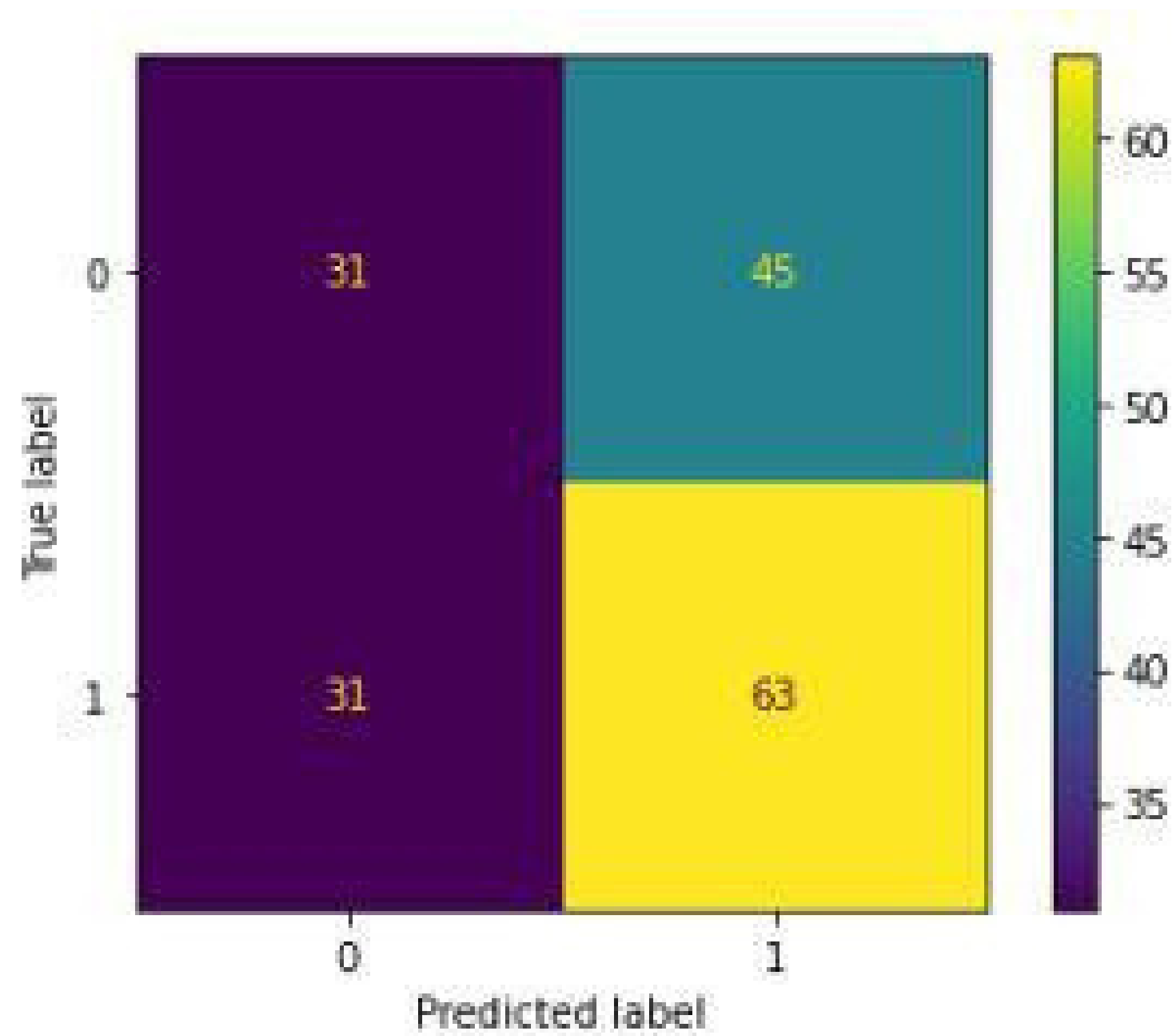




we have trained **XGBClassifier** has the highest performance but it is prone to overfitting as the difference between the training and the validation accuracy is too high. But in the case of the Logistic Regression, this is not the case. Now let's plot a confusion matrix for the validation data.

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

# OUTPUT



# CONCLUSION

**A stock price is a given for every share issued by a publicly-traded company. The price is a reflection of the company's value – what the public is willing to pay for a piece of the company.**

