```python
import yfinance as yf
import pandas as pd
import numpy as np

banknifty = yf.Ticker("^NSEBANK")
df = banknifty.history(period="max")

df = df.drop(["Dividends", "Stock Splits"], axis=1, errors="ignore")
df.head()
```

```
                                  Open         High          Low
Close  \
Date

2007-09-17 00:00:00+05:30  6897.919911  6977.119187  6842.920550
6897.020020
2007-09-18 00:00:00+05:30  6921.069480  7078.867940  6883.520112
7059.567871
2007-09-19 00:00:00+05:30  7110.917439  7419.263957  7110.917439
7401.764160
2007-09-20 00:00:00+05:30  7404.864086  7462.813119  7343.514701
7390.063965
2007-09-21 00:00:00+05:30  7378.213894  7506.262696  7367.064122
7464.413086

                           Volume
Date
2007-09-17 00:00:00+05:30       0
2007-09-18 00:00:00+05:30       0
2007-09-19 00:00:00+05:30       0
2007-09-20 00:00:00+05:30       0
2007-09-21 00:00:00+05:30       0
```

```python
# Daily return
df["Return"] = df["Close"].pct_change()

# Moving averages
df["MA5"] = df["Close"].rolling(5).mean()
df["MA10"] = df["Close"].rolling(10).mean()

# Momentum
df["Momentum"] = df["Close"] - df["Close"].shift(5)

# Volatility
df["Volatility"] = df["Return"].rolling(5).std()

df.dropna(inplace=True)

# 1 = BUY (price goes up tomorrow)
# 0 = DO NOT BUY
df["Target"] = (df["Return"].shift(-1) > 0).astype(int)
```

```
df.dropna(inplace=True)

df[["Return", "Target"]].head()
```

Date	Return	Target
2007-09-28 00:00:00+05:30	0.026622	0
2007-10-01 00:00:00+05:30	-0.006802	1
2007-10-03 00:00:00+05:30	0.013822	0
2007-10-04 00:00:00+05:30	-0.007656	0
2007-10-05 00:00:00+05:30	-0.023725	0

```
features = ["Return", "MA5", "MA10", "Momentum", "Volatility"]

X = df[features]
y = df["Target"]

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False
)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled, y_train)

LogisticRegression(max_iter=1000)

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

y_pred = model.predict(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test,
y_pred))

Accuracy: 0.533969010727056

Confusion Matrix:
 [[ 76 306]
```

```
 [ 85 372]]

Classification Report:
              precision    recall  f1-score   support

           0       0.47      0.20      0.28       382
           1       0.55      0.81      0.66       457

    accuracy                           0.53       839
   macro avg       0.51      0.51      0.47       839
weighted avg       0.51      0.53      0.48       839


latest_data = X.iloc[-1:].values
latest_scaled = scaler.transform(latest_data)

decision = model.predict(latest_scaled)[0]
probability = model.predict_proba(latest_scaled)[0][1]

if decision == 1:
    print(f"□ BUY SIGNAL for tomorrow (Confidence:
{probability:.2f})")
else:
    print(f"□ NO BUY signal for tomorrow (Confidence: {1 -
probability:.2f})")

□ BUY SIGNAL for tomorrow (Confidence: 0.52)

C:\Users\chait\AppData\Local\Programs\Orange\Lib\site-packages\
sklearn\base.py:465: UserWarning: X does not have valid feature names,
but StandardScaler was fitted with feature names
  warnings.warn(

# Number of past days to use
LAGS = 5

for i in range(1, LAGS + 1):
    df[f"Lag_{i}"] = df["Close"].shift(i)

# Target = tomorrow's price
df["Target"] = df["Close"].shift(-1)

df.dropna(inplace=True)
df.head()

                                  Open         High          Low
Close  \
Date

2007-10-08 00:00:00+05:30  7853.058393  7935.357727  7516.362609
7626.311035
```

```
2007-10-09 00:00:00+05:30  7580.811767  7916.358159  7534.962202
7895.758301
2007-10-10 00:00:00+05:30  7960.557454  8080.955958  7907.258268
8030.556641
2007-10-11 00:00:00+05:30  8054.206291  8177.655053  8005.407053
8158.705078
2007-10-12 00:00:00+05:30  8093.555760  8132.005508  7889.708034
7933.907715

                            Volume    Return          MA5         MA10
\
Date

2007-10-08 00:00:00+05:30       0 -0.027896  7918.497852  7850.383691

2007-10-09 00:00:00+05:30       0  0.035331  7900.168066  7874.878418

2007-10-10 00:00:00+05:30       0  0.017072  7886.718262  7915.027979

2007-10-11 00:00:00+05:30       0  0.015958  7911.297949  7955.317529

2007-10-12 00:00:00+05:30       0 -0.027553  7929.047754  7965.352393


                              Momentum  Volatility       Target
Lag_1  \
Date

2007-10-08 00:00:00+05:30 -415.795898    0.016514  7895.758301
7845.158691
2007-10-09 00:00:00+05:30  -91.648926    0.026545  8030.556641
7626.311035
2007-10-10 00:00:00+05:30  -67.249023    0.027065  8158.705078
7895.758301
2007-10-11 00:00:00+05:30  122.898438    0.027746  7933.907715
8030.556641
2007-10-12 00:00:00+05:30   88.749023    0.028716  8286.203125
8158.705078

                                   Lag_2        Lag_3        Lag_4
Lag_5
Date

2007-10-08 00:00:00+05:30  8035.806641  8097.805664  7987.407227
8042.106934
2007-10-09 00:00:00+05:30  7845.158691  8035.806641  8097.805664
7987.407227
2007-10-10 00:00:00+05:30  7626.311035  7845.158691  8035.806641
8097.805664
2007-10-11 00:00:00+05:30  7895.758301  7626.311035  7845.158691
```

```
8035.806641
2007-10-12 00:00:00+05:30  8030.556641  7895.758301  7626.311035
7845.158691
```

```
from sklearn.model_selection import train_test_split

X = df[[f"Lag_{i}" for i in range(1, LAGS + 1)]]
y = df["Target"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False
)
```

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
from sklearn.metrics import mean_absolute_error

y_pred = model.predict(X_test)
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
MAE: 463.9822392415442
```

```
latest_lags = X.iloc[-1:].values
tomorrow_price = model.predict(latest_lags)[0]

print("□ Predicted price for tomorrow:", round(tomorrow_price, 2))
```

```
□ Predicted price for tomorrow: 58969.36
```

```
C:\Users\chait\AppData\Local\Programs\Orange\Lib\site-packages\
sklearn\base.py:465: UserWarning: X does not have valid feature names,
but LinearRegression was fitted with feature names
  warnings.warn(
```

```
# this is just the predicted closing price on 1

# Create return
df["Return"] = df["Close"].pct_change()

# Target = tomorrow's return
df["Target_Return"] = df["Return"].shift(-1)

df.dropna(inplace=True)

LAGS = 5
for i in range(1, LAGS + 1):
    df[f"Return_Lag_{i}"] = df["Return"].shift(i)
```

```
df.dropna(inplace=True)

X = df[[f"Return_Lag_{i}" for i in range(1, LAGS + 1)]]
y = df["Target_Return"]

from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(
    n_estimators=500,
    max_depth=6,
    min_samples_leaf=20,
    random_state=42
)
model.fit(X_train, y_train)
```

```
RandomForestRegressor(max_depth=6, min_samples_leaf=20,
n_estimators=500,
                      random_state=42)
```

```
df["MA5"] = df["Close"].rolling(5).mean()
df["MA10"] = df["Close"].rolling(10).mean()
df["Volatility"] = df["Return"].rolling(5).std()
df["Momentum"] = df["Close"] - df["Close"].shift(5)

predictions = []
actuals = []

for i in range(2000, len(X)):
    X_train = X.iloc[:i]
    y_train = y.iloc[:i]
    X_test = X.iloc[i:i+1]
    y_test = y.iloc[i]

    model.fit(X_train, y_train)
    pred = model.predict(X_test)[0]

    predictions.append(pred)
    actuals.append(y_test)

pred_return = model.predict(latest_X)[0]
vol = df["Volatility"].iloc[-1]

upper = last_close * (1 + pred_return + vol)
lower = last_close * (1 + pred_return - vol)
```