```
import streamlit as st

import yfinance as yf

from sklearn.ensemble import RandomForestRegressor

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error, mean_squared_error

import pandas as pd


st.title("Coca-Cola Stock Price Prediction")


# Define ticker

ticker = "KO"


# ---------------------------

# Download historical data

# ---------------------------

data = yf.download(ticker, start="2015-01-01")


# Flatten MultiIndex if present

if isinstance(data.columns, pd.MultiIndex):

    data.columns = [

        col[0] if col[1] == "" else f"{col[0]}_{col[1]}" for col in data.columns

    ]


# Remove ticker suffix (_KO) to keep simple names

data.columns = [col.replace(f"_{ticker}", "") for col in data.columns]


# ---------------------------

# Feature Engineering

# ---------------------------

data["MA_20"] = data["Close"].rolling(20).mean()

data["MA_50"] = data["Close"].rolling(50).mean()
```

```
data["Daily_Return"] = data["Close"].pct_change()
data["Volatility"] = data["Daily_Return"].rolling(20).std()
data = data.dropna()

# ---------------------------
# Train/Test split
# ---------------------------
features = ["Open", "High", "Low", "Volume", "MA_20", "MA_50", "Daily_Return", "Volatility"]
target = "Close"

X = data[features]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False, random_state=42
)

# Train model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predictions on test
y_pred = model.predict(X_test)

# Metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

st.subheader("Model Performance")
st.write("MAE:", mae)
st.write("MSE:", mse)
```

```
# ---------------------------
# Plot chart
# ---------------------------
st.subheader("Historical Prices with Moving Averages")
st.line_chart(data[["Close", "MA_20", "MA_50"]])

# ---------------------------
# Live prediction
# ---------------------------
live_data = yf.download(ticker, period="1d", interval="1m")

# Flatten MultiIndex if present
if isinstance(live_data.columns, pd.MultiIndex):
    live_data.columns = [
        col[0] if col[1] == "" else f"{col[0]}_{col[1]}" for col in live_data.columns
    ]

# Remove ticker suffix (_KO)
live_data.columns = [col.replace(f"_{ticker}", "") for col in live_data.columns]

# Feature engineering for live data
live_data["MA_20"] = live_data["Close"].rolling(20).mean()
live_data["MA_50"] = live_data["Close"].rolling(50).mean()
live_data["Daily_Return"] = live_data["Close"].pct_change()
live_data["Volatility"] = live_data["Daily_Return"].rolling(20).std()
live_data.fillna(0, inplace=True)

# Latest row for prediction
latest_features = live_data[features].iloc[-1:]
live_prediction = model.predict(latest_features)
```

```
st.subheader("Live Prediction")

st.write(f"Predicted Closing Price: {live_prediction[0]}")
```