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]}")