Program to demonstrate stack price prediction using regression model with ML.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
np.random.seed(42)
days = np.arange(1, 101)
price = 50 + 0.5 * days + np.random.normal(0, 2, size=100)
df = pd.DataFrame({"day": days, "price": price})
print("First 5 rows of dataset:\n", df.head())
X = df[['day']]
y = df['price']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(
X_scaled, y, test_size=0.2, random_state=42
)
rf_model=RandomForestRegressor(n_estimators=10, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
print("\n Random Forest Results:")
print("MSE:", mean_squared_error(y_test, y_pred_rf))
print("R2 Score:", r2_score(y_test, y_pred_rf))
plt.scatter(X_test, y_test, color='blue', label="Actual")
plt.scatter(X_test, y_pred_rf, color='red', label="RF Predicted")
plt.title("Random Forest Predictions vs Actual")
plt.xlabel("Day (scaled)")
plt.ylabel("Stock Price")
plt.legend()
plt.show()
y_test_array = np.array(y_test)
sorted_idx = np.argsort(X_test.flatten())
plt.figure(figsize=(8,5))
plt.plot(X_test.flatten()[sorted_idx], y_test_array[sorted_idx], label="Actual", color="black")
plt.plot(X_test.flatten()[sorted_idx], y_pred_rf[sorted_idx], label="RF Predicted", color="red")
plt.title("Stock Price Prediction (Random Forest)")
plt.xlabel("Day (scaled)")
plt.ylabel("Price")
plt.legend()
plt.show()
```

Program to demonstrate stack price prediction using regression model with DL.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
np.random.seed(42)
days = np.arange(1, 101)
price = 50 + 0.5 * days + np.random.normal(0, 2, size=100)
df = pd.DataFrame({"day": days, "price": price})
print("First 5 rows of dataset:\n", df.head())
X = df[['day']]
y = df['price']
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
X_scaled, y, test_size=0.2, random_state=42
)
dl_model = Sequential([
Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
Dropout(0.2),
Dense(32, activation='relu'),
Dense(1) # Regression output (no activation)
])
dl_model.compile(optimizer='adam', loss='mse', metrics=['mae'])
history = dl_model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=100, batch_size=8, verbose=0)
dl_mse, dl_mae = dl_model.evaluate(X_test, y_test, verbose=0)
print("\n Deep Learning Results:")
print("MSE:", dl_mse)
print("MAE:", dl_mae)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.title("DL Model Training Loss (MSE)")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
y_pred_dl = dl_model.predict(X_test).flatten()
y_test_array = np.array(y_test)
sorted_idx = np.argsort(X_test.flatten())
plt.figure(figsize=(8,5))
plt.plot(X_test.flatten()[sorted_idx], y_test_array[sorted_idx], label="Actual", color="black")
plt.plot(X_test.flatten()[sorted_idx], y_pred_dl[sorted_idx], label="DL Predicted",
color="green")
plt.title("Stock Price Prediction with Deep Learning")
plt.xlabel("Day (scaled)")
plt.ylabel("Price")
plt.legend()
plt.show()
```