Financial forecasting: Gold prices and investment options

Libraries Overview

This slide explains the main Python libraries used in the code.

- pandas:

Used to create and manage tabular data (DataFrame).

- numpy:

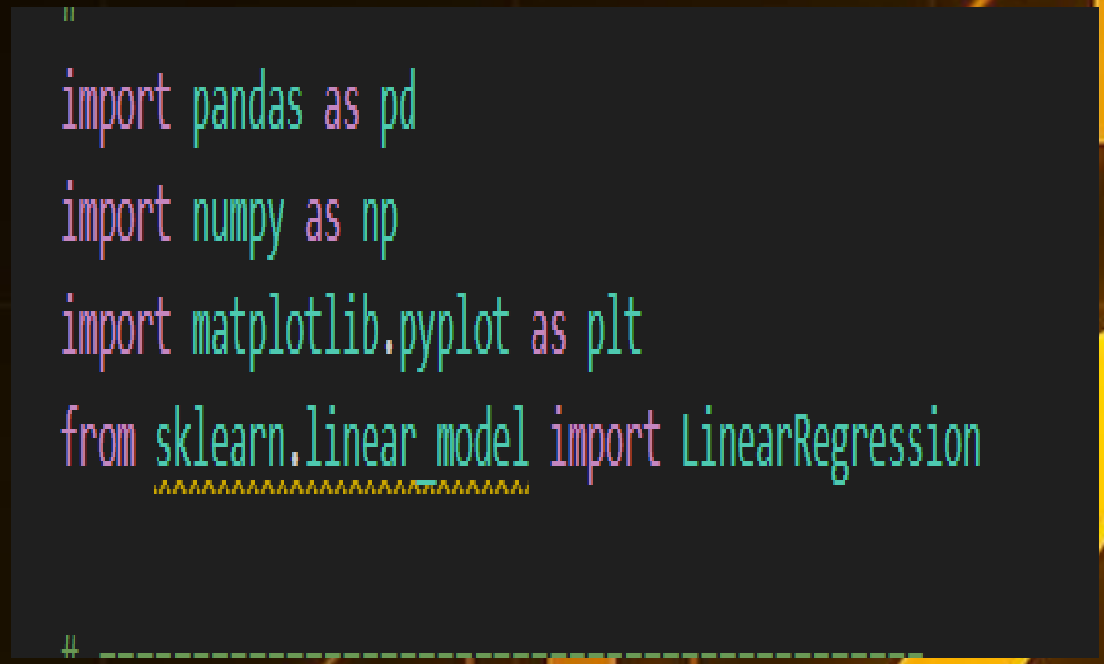
Used for numerical operations and creating arrays.

- matplotlib.pyplot:

Used for plotting and visualizing data.

- sklearn.linear_model.LinearRegression:

Used to build a regression model to predict tomorrow's price.



Loading the Data

The gold prices for 30 days are stored inside a Python list.

Then a pandas DataFrame is created with two columns:

- Day:

Numbers from 1 to 30.

- Price:

The actual gold price for each day.

This structure allows easy processing for machine learning and plotting.

```
prices = [
    174850, 175100, 175300, 175500, 175700, 175950,
    176100, 176200, 176300, 176400, 176500, 176600,
    176700, 176800, 176850, 176900, 176950, 177000,
    177050, 177100, 177120, 177140, 177160, 177180,
    177190, 177200, 177210, 177215, 177220, 177224
]

df = pd.DataFrame({
    "Day": np.arange(1, 31),
    "Price": prices
})
```

Preparing Data for Regression

In this step, the code selects data for training the model:

- X contains the 'Day' column — the input feature.

- y contains the 'Price' column — the target values.

LinearRegression() is then created and trained using model.fit(X, y).

```
next_day = np.array([[31]])
predicted_price = model.predict(next_day)[0]
current_price = df["Price"].iloc[-1]

print("Current gold price (per gram):", f"{current_price:,.0f} IQD")
print("Predicted gold price for tomorrow (per gram):", f"{predicted_price:,.2f} IQD")
```

Predicting Tomorrow's Gold Price

After training, the model predicts the price for day 31 using:

model.predict([[31]]).

It also reads the current (last) day's price to compare and decide if investment is good.

```
next_day = np.array([[31]])
predicted_price = model.predict(next_day)[0]
current_price = df["Price"].iloc[-1]

print("Current gold price (per gram):", f"{current_price:,.0f} IQD")
print("Predicted gold price for tomorrow (per gram):", f"{predicted_price:,.2f} IQD")
```

Plotting the Results

matplotlib is used to draw:

- The actual prices for 30 days.
- A dashed horizontal line at the predicted price for day 31.

A text label appears on the plot showing the recommendation (invest or not).

```
plt.plot(df["Day"], df["Price"], marker='o')
plt.axhline(y=predicted_price, linestyle='--')

advice_text = "⚠️ Do not invest today" if predicted_price < current_price else "✅ Investment may be good"
color = "red" if predicted_price < current_price else "green"

plt.text(30.5, predicted_price, advice_text, color=color)

plt.xlabel("Day")
plt.ylabel("Gold Price (IQD)")
plt.title("Gold Price Prediction")
plt.show()
```

Final Decision Logic

The code compares predicted_price with current_price:

- If predicted > current → Good investment.
- If predicted < current → Do not invest today.

If the investment is good, the program asks how many grams the user wants to buy and calculates the expected profit.

```
if predicted_price < current_price:
    print("⚠️ Advice: The predicted price tomorrow is lower than today, better not to invest today.")
else:
    print("✅ Advice: The predicted price tomorrow is higher than today, investment may be a good option.")

    grams = float(input("Enter the number of grams you want to invest in: "))
    profit_per_gram = predicted_price - current_price
    total_profit = profit_per_gram * grams
    total_value = (current_price * grams) + total_profit

    print("Profit per gram:", f"{profit_per_gram:,.2f} IQD")
    print("Expected profit from your investment:", f"{total_profit:,.2f} IQD")
    print("Total investment value tomorrow:", f"{total_value:,.2f} IQD")
```

Thank you for listening.

Mustafa Anwar Fakher , Mohammed Ali Jassim Mohammed & Yahya Raed Jabbar