Experiment 2: Predicting gold prices using Linear Regression Model involves data collection, preprocessing, model training, and evaluation. Below is a detailed example using Python and Scikit-learn to predict gold prices.

Step 1: Import Libraries

```
#pip install yfinance this library is need for finance estimations
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import yfinance as yf

```

Step 2: Data Collection: use historical gold price data from Yahoo Finance. You can install the yfinance library if you haven't already by running "pip install yfinance"

```
# Download historical gold price data
gold_data = yf.download('GC=F', start='2010-01-01', end='2023-01-01')

# Display the first few rows of the dataset
print(gold_data.head())

```

[6] ✓ 0.1s Python

```
[**********100%%**********]  1 of 1 completed
                 Open         High          Low        Close    Adj Close  \
Date
2010-01-04  1117.699951  1122.300049  1097.099976  1117.699951  1117.699951
2010-01-05  1118.099976  1126.500000  1115.000000  1118.099976  1118.099976
2010-01-06  1135.900024  1139.199951  1120.699951  1135.900024  1135.900024
2010-01-07  1133.099976  1133.099976  1129.199951  1133.099976  1133.099976
2010-01-08  1138.199951  1138.199951  1122.699951  1138.199951  1138.199951
```

Step 3: Data Pre-processing: We'll use the 'Close' prices for prediction and create a feature set based on the date index.

```
# Use 'Close' price as the target variable
gold_data = gold_data[['Close']]
gold_data = gold_data.dropna()  # Drop any rows with NaN values

# Reset index to use the date as a feature
gold_data.reset_index(inplace=True)

# Convert 'Date' to numerical format
gold_data['Date'] = gold_data['Date'].map(pd.Timestamp.toordinal)

# Display the first few rows after preprocessing
print(gold_data.head())

```

[7] ✓ 0.0s Python

```
     Date        Close
0  733776  1117.699951
1  733777  1118.099976
```

Step 4: Split Data into Training and Testing Sets

```
# Define features (X) and target (y)
X = gold_data[['Date']]
y = gold_data['Close']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```

] ✓ 0.0s Python

Step 5: Create and Train the Model

```
# Create the model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)

```

[9] ✓ 0.0s

```
LinearRegression()
```

Step 6: Make Predictions

```
# Make predictions
y_pred = model.predict(X_test)

```

✓ 0.0s

Step 7: Evaluate the Model

```
# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)

```

] ✓ 0.0s Python

```
Mean Squared Error (MSE): 51075.211915630076
R² Score: 0.15022079688041157
```

Step 8: Visualize the Results

```
# Plot the results
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='blue', label='Actual Prices')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted Prices')
plt.xlabel('Date')
plt.ylabel('Gold Price')
plt.title('Gold Price Prediction using Linear Regression')
plt.legend()
plt.show()

```

✓ 0.3s