Visualisasi data harga emas pada Regresi Linier

Data harus divisualisasikan dalam bentuk grafis ataupun data korelasi. Tujuan dari visualisasi tersebut adalah mempermudah dalam menentukan apakah data tersebut cocok dilakukan prediksi harga emas dengan metode regresi linier atau tidak.

Parameter Fitur metode regresi linier

Pemilihan Fitur didasarkan pada fitur apa yang dipilih yang nantinya akan dilakukan train dan test pada harga emas. Berikut adalah detail parameter yang digunakan :

Tabel 1 Parameter data harga emas

Kolom	Jenis	Keterangan
DATE	Tanggal	Tanggal Kurs
SPX	Numerik	Nilai tukar kurs dalam SPX
GLD	Numerik	Nilai tukar kurs dalam GLD
USO	Numerik	Nilai tukar kurs dalam USO
SLV	Numerik	Nilai tukar kurs dalam SLV
EUR/USD	Numerik	Nilai tukar kurs dalam EUR atau USD

1. Fitur 1 variabel

   Fitur yang digunakan adalah “SLV” sebagai variabel x, dan “GLD” sebagai variabel y.
2. Fitur Multivariabel

   Fitur yang digunakan adalah seluruh fitur selain “GLD” yang sebagai variabel y, dan “DATE”.

Regresi linier pada prediksi harga emas

1. Import library

```
# Library for Data Analysis
import pandas as pd
```

```
import numpy as np

# Library for Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt

# Library for Scaling Data
from sklearn.preprocessing import StandardScaler,MinMaxScaler, RobustScaler

# Library for Creating Model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

2. Import Data

```
df =
pd.read_excel("https://docs.google.com/spreadsheets/d/1vS-HnK4E9ykaMyL-zfH3esbVd-OuvAbUE1YqLxjtfeA/export?gid=0&format=xlsx")

df.head()
df.describe()
df.info
```

3. Data Cleaning

```
df.isna().sum()
df.duplicated(keep=False).sum()
df.boxplot()
```

Outliers Detection

```
def outlier(column):
  Q1 = np.percentile(column, 25)
  Q3 = np.percentile(column, 75)

  IQR = Q3 - Q1

  IQR_min = Q1 - (1.5 * IQR)
  IQR_max = Q3 + (1.5 * IQR)

  outlier = []
  for x in column:
    if (x < IQR_min or x > IQR_max):
       outlier.append(x)

  return outlier
```

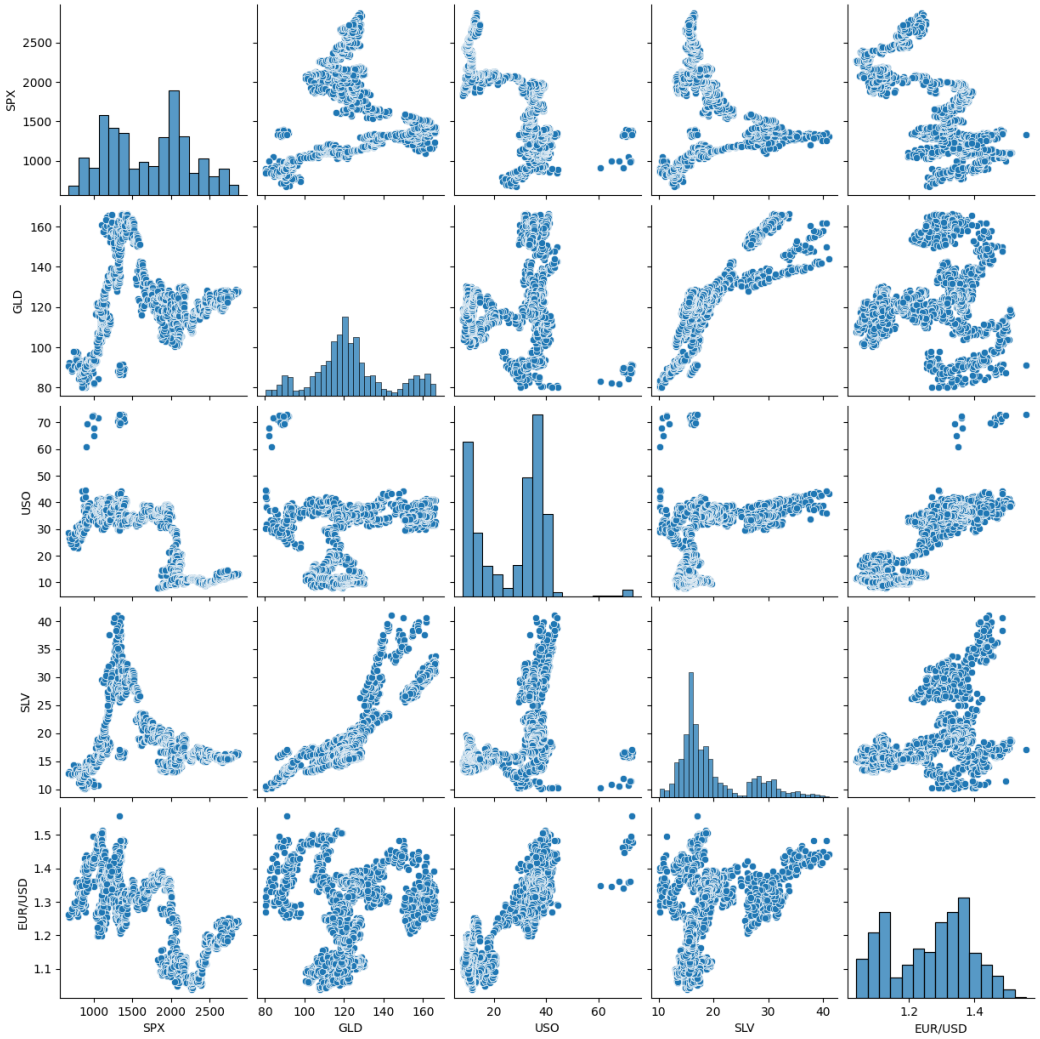
Outliers Cleaning

```
def clean_outlier(df, column):
  Q1 = np.percentile(df[column], 25)
  Q3 = np.percentile(df[column], 75)

  IQR = Q3 - Q1

  IQR_min = Q1 - (1.5 * IQR)
  IQR_max = Q3 + (1.5 * IQR)

  return df.loc[(df[column] >= IQR_min) & (df[column] <= IQR_max)]

old_length = len(df)
```

```
df = clean_outlier(df, "USO")
df = clean_outlier(df, "SLV")
df = clean_outlier(df, "EUR/USD")
df = clean_outlier(df, "GLD")
new_length = len(df)
print(f"Amount of Data Decreasing from %d to %d" % (old_length, new_length))
```

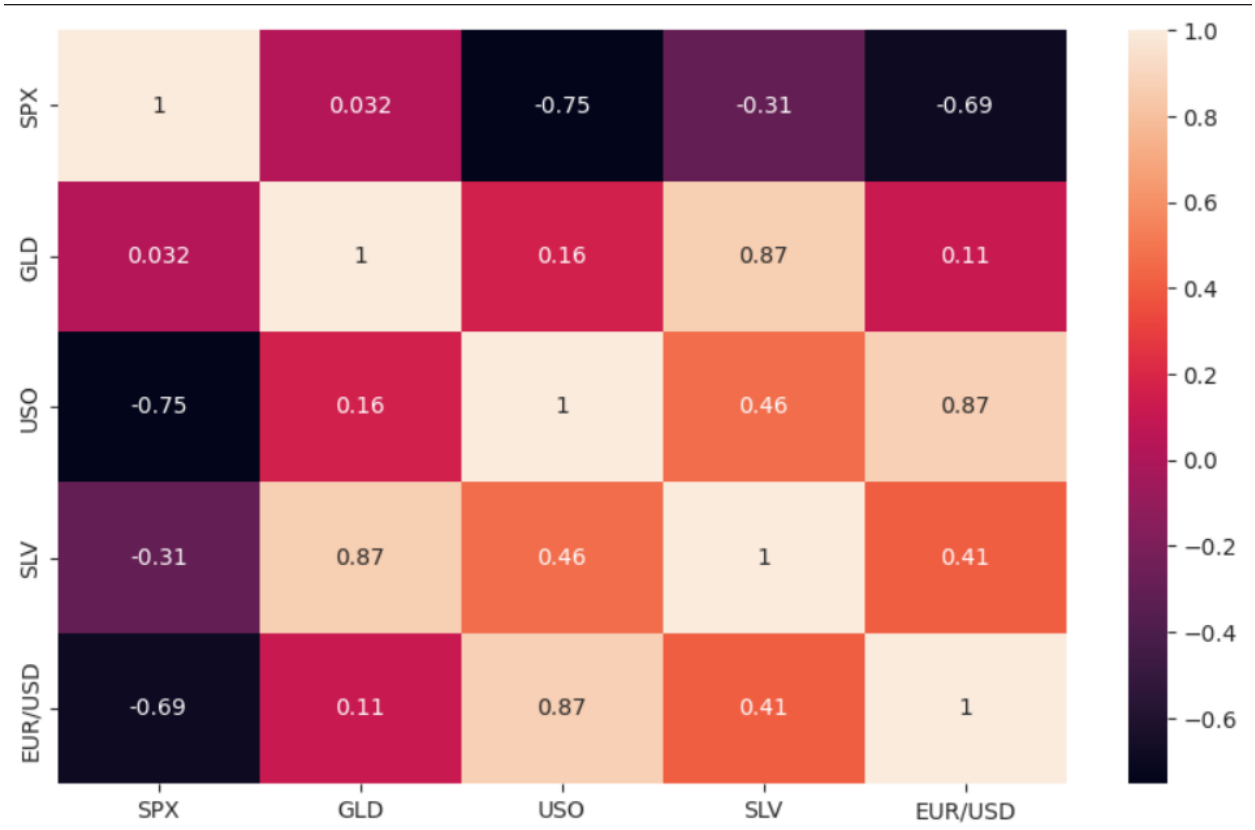
4. Visualisasi Data

```
sns.pairplot(df)
```



5. Korelasi Data

```
corr_data = df.corr()
plt.figure(figsize=(10,6))
sns.set_style('ticks')

sns.heatmap(corr_data, annot=True)
plt.show()
```



6. Preprocessing data

```
scaler = StandardScaler()
```

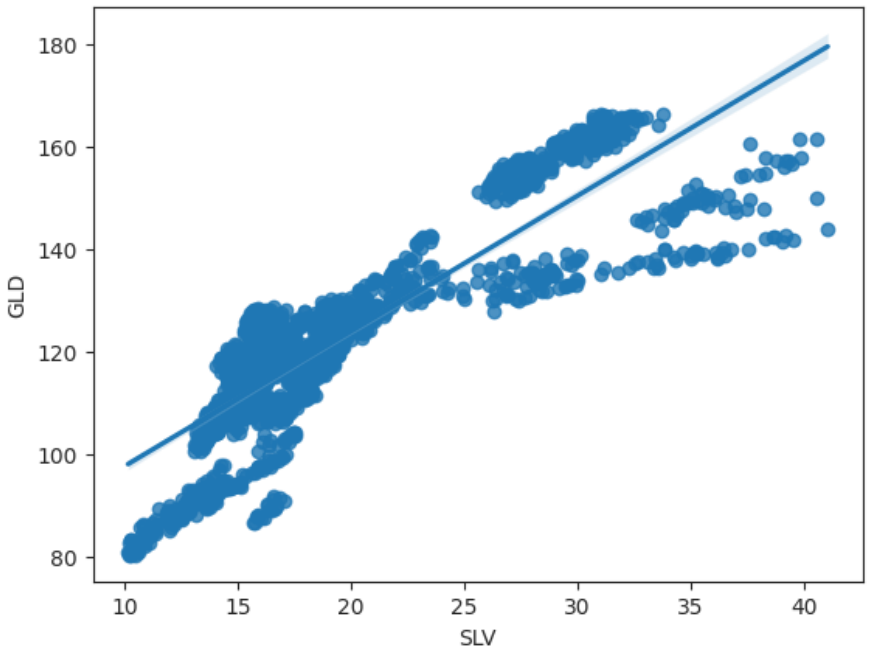
```
# Single
X_single = scaler.fit_transform(df['SLV'].values.reshape(-1, 1))


# Quadratic X
```

```
X_quadratic = df['SLV']
X_quadratic = pd.DataFrame(X_quadratic)
X_quadratic['SLV2'] = X_quadratic['SLV']**2
X_quadratic['SLV3'] = X_quadratic['SLV']**3
quadratic_column = X_quadratic.columns
X_quadratic = scaler.fit_transform(X_quadratic)

# Multi
features=df.drop(['GLD', 'Date'], axis=1)
X_mult=scaler.fit_transform(features)
mult_column=features.columns

y = df['GLD']
```

7. Single Variable

   Pemilihan Fitur

```
sns.regplot(x='SLV', y='GLD', data=df)
```



Split data

```
X_train_single, X_test_single, y_train_single, y_test_single =
train_test_split(X_single, y,test_size=0.2, random_state=50,)
```

```
X_train_single.shape
```

```
X_test_single.shape
```

Data Training

```
regressor_single = LinearRegression()
regressor_single.fit(X_train_single, y_train_single)
print("Intercept dari persamaan yang didapat adalah = %f" %
regressor_single.intercept_)
print("Coefisien dari persamaan yang didapat adalah = %f" % regressor_single.coef_)
```

Data Testing

```
y_pred_train_single = regressor_single.predict(X_train_single)
y_pred_test_single = regressor_single.predict(X_test_single)
```

```
regressor_single.score(X_train_single, y_train_single)
```

```
regressor_single.score(X_test_single, y_test_single)
```

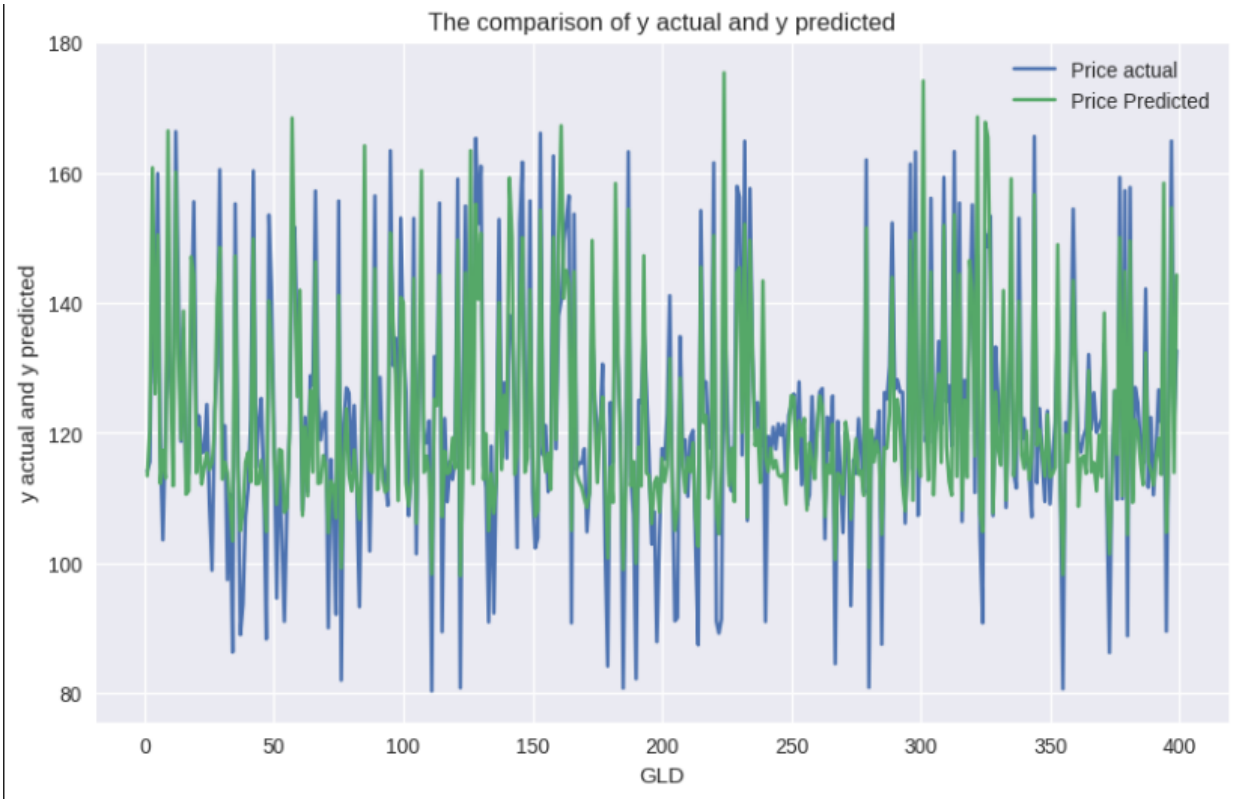
Comparison Result

```
prediksi_single=pd.DataFrame({'Y Actual': y_test_single, 'Y Predicted':
y_pred_test_single})
prediksi_single.head()
```

```
plt.style.use('seaborn')
fig = plt.figure(figsize=(10,6))
ax = plt.axes()
```

```
ax.plot(np.arange(1,X_test_single.shape[0]+1,1), prediksi_single['Y Actual'],
label='Price actual')
ax.plot(np.arange(1,X_test_single.shape[0]+1,1), prediksi_single['Y Predicted'],
label='Price Predicted')
ax.set_title('The comparison of y actual and y predicted')
ax.set_ylabel('y actual and y predicted')
ax.set_xlabel('GLD')
ax.legend()

plt.show()
```

Hasil :



Matrik Error (Train) :

Keterangan	Nilai

Mean Absolute Error (MAE)	7.509027622375216
Mean Squared Error (MSE)	90.53208247409555
Root Mean Squared Error (RMSE)	9.514834863206799
Max Error	36
R2 Score	0.75186

Matriks Error (Test) :

Keterangan	Nilai
Mean Absolute Error (MAE)	7.691890560229349
Mean Squared Error (MSE)	93.4856232180074
Root Mean Squared Error (RMSE)	9.668796368628692
Max Error	33
R2 Score	0.73744

8. Single Variabel Kuadrat

Data Training

```
X_train_quadrat, X_test_quadrat, y_train_quadrat, y_test_quadrat =
train_test_split(X_quadratic, y, test_size=0.2, random_state=50,)
```

```
X_train_quadrat.shape
```

```
X_test_quadrat.shape
```

```
coef=regressor_quadrat.coef_
coef
```

```
coeff_quadrat = pd.DataFrame(coef.reshape(-1,1),
                             quadratic_column,
                             columns=['Slope']
```

```
                    )
coeff_quadrat.sort_values('Slope')
```

Hasil :

1. Intercept : 122.77840256369024
2. Coefisien : [44.09982684, -20.21279244, -7.77706233]

Data Testing

```
y_pred_train_quadrat = regressor_quadrat.predict(X_train_quadrat)
y_pred_test_quadrat = regressor_quadrat.predict(X_test_quadrat)
```

```
regressor_quadrat.score(X_train_quadrat, y_train_quadrat)
```

```
regressor_quadrat.score(X_test_quadrat, y_test_quadrat)
```

Hasil :

1. Train : 0.817899023608716
2. Test : 0.8251063371804406

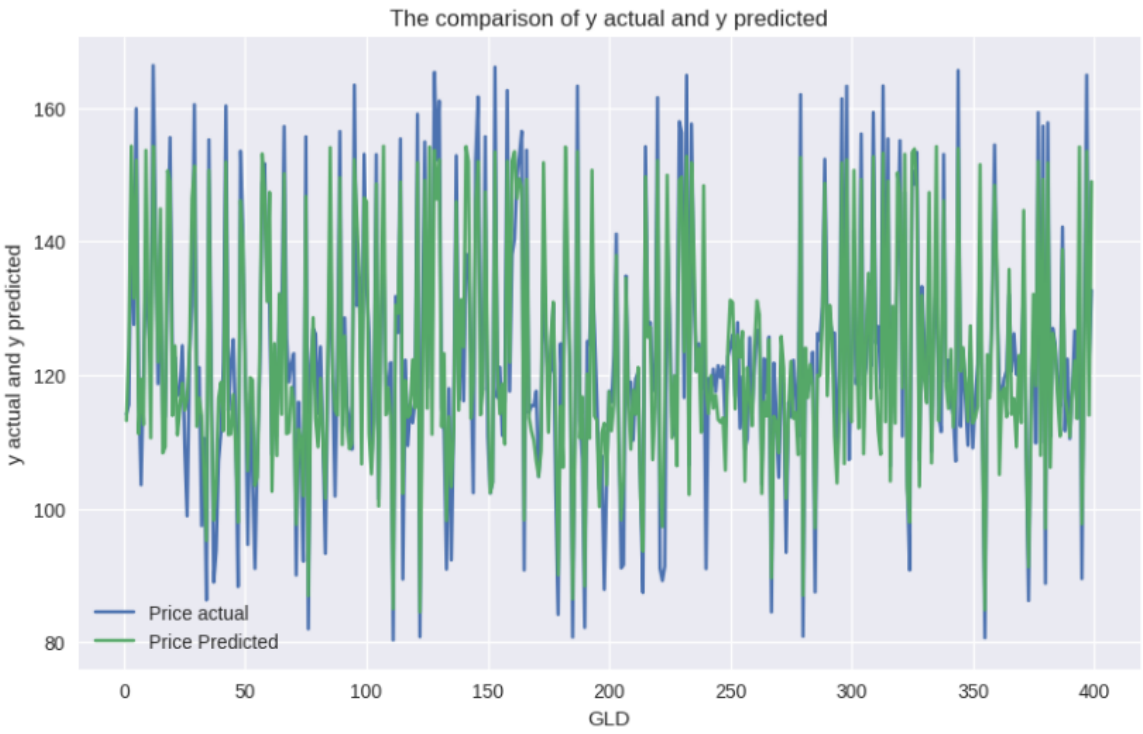
Comparison Result

```
prediksi_quadrat=pd.DataFrame({'Y Actual': y_test_quadrat, 'Y Predicted':
y_pred_test_quadrat})
prediksi_quadrat.head()
```

```
plt.style.use('seaborn')
fig = plt.figure(figsize=(10,6))
ax = plt.axes()

ax.plot(np.arange(1,X_test_quadrat.shape[0]+1,1), prediksi_quadrat['Y Actual'],
label='Price actual')
ax.plot(np.arange(1,X_test_quadrat.shape[0]+1,1), prediksi_quadrat['Y Predicted'],
```

```
label='Price Predicted')
ax.set_title('The comparison of y actual and y predicted')
ax.set_ylabel('y actual and y predicted')
ax.set_xlabel('GLD')
ax.legend()

plt.show()
```

Hasil :



Matrik Error (Train) :

Keterangan	Nilai
Mean Absolute Error (MAE)	6.581660316936941
Mean Squared Error (MSE)	66.43749527369637
Root Mean Squared Error (RMSE)	8.150919903526004
Max Error	27

R2 Score	0.8179

Matriks Error (Test) :

Keterangan	Nilai
Mean Absolute Error (MAE)	6.469994637715179
Mean Squared Error (MSE)	62.27199534068499
Root Mean Squared Error (RMSE)	7.89126069400099
Max Error	27
R2 Score	0.82511

9. Multi Variable

Split Data

```
X_train_mult, X_test_mult, y_train_mult, y_test_mult = train_test_split(X_mult, y,
test_size=0.2, random_state=50, )
```

```
X_train_mult.shape
```

```
X_test_mult.shape
```

Data Training

```
regressor_mult = LinearRegression()
regressor_mult.fit(X_train_mult, y_train_mult)
```

```
intercept=regressor_mult.intercept_
intercept
```

```
coef=regressor_mult.coef_
coef
```

```
coeff_mult = pd.DataFrame(coef.reshape(-1,1),
```

```
                mult_column,
                columns=['Slope']
                )
coeff_mult.sort_values('Slope')
```

Hasil :

1. Intercept : 122.81417704795611
2. Coefisien : [4.98269239, 0.0597284 , 18.77564371, -2.05144249]

Data Testing

```
y_pred_train_mult = regressor_mult.predict(X_train_mult)
y_pred_test_mult = regressor_mult.predict(X_test_mult)
```

```
regressor_mult.score(X_train_mult, y_train_mult)
```

```
regressor_mult.score(X_test_mult, y_test_mult)
```

Hasil :

3. Train : 0.853157291710249
4. Test : 0.8444564178520869

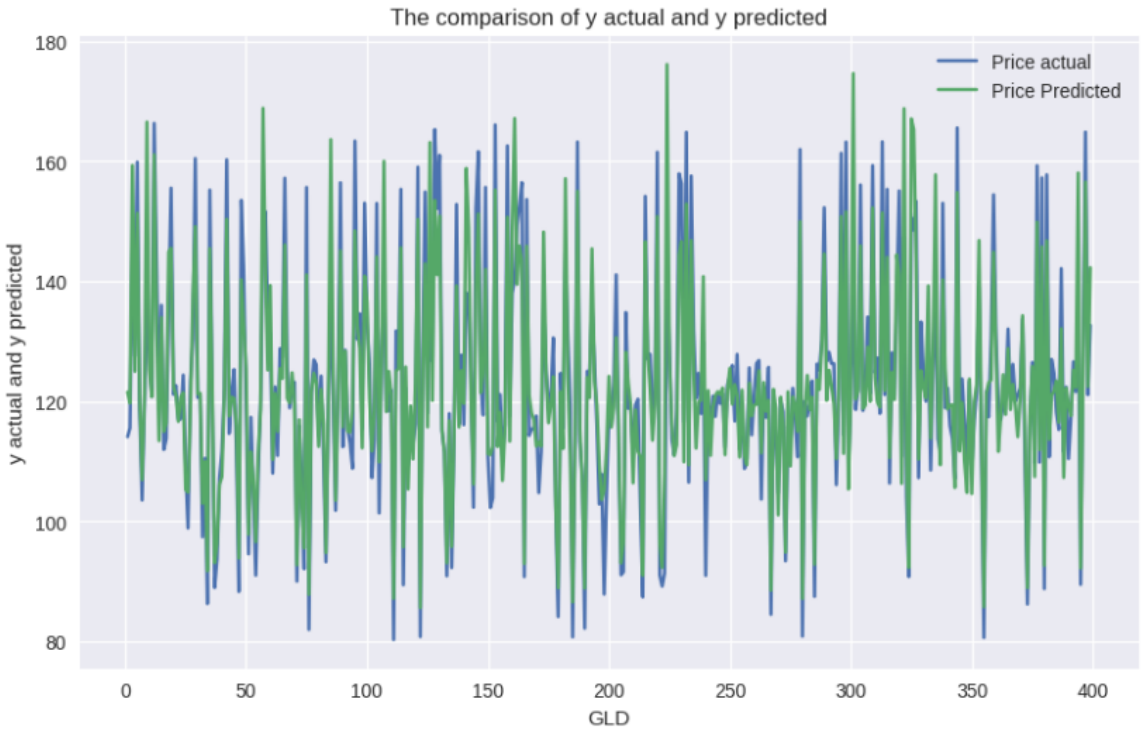
Comparison Result

```
prediksi_mult=pd.DataFrame({'Y Actual': y_test_mult, 'Y Predicted':
y_pred_test_mult})
prediksi_mult.head()
```

```
plt.style.use('seaborn')
fig = plt.figure(figsize=(10,6))
ax = plt.axes()

ax.plot(np.arange(1,X_test_mult.shape[0]+1,1), prediksi_mult['Y Actual'], label='Price
```

```
actual')
ax.plot(np.arange(1,X_test_mult.shape[0]+1,1), prediksi_mult['Y Predicted'],
label='Price Predicted')
ax.set_title('The comparison of y actual and y predicted')
ax.set_ylabel('y actual and y predicted')
ax.set_xlabel('GLD')
ax.legend()

plt.show()
```

Hasil :



Matrik Error (Train) :

Keterangan	Nilai
Mean Absolute Error (MAE)	5.448217519592641
Mean Squared Error (MSE)	53.57391229476163

Root Mean Squared Error (RMSE)	7.319420215752176
Max Error	37
R2 Score	0.85316

Matriks Error (Test) :

Keterangan	Nilai
Mean Absolute Error (MAE)	5.413084709570092
Mean Squared Error (MSE)	55.38227667392104
Root Mean Squared Error (RMSE)	7.441926946290257
Max Error	34
R2 Score	0.84466