

Presentasi UTS Machine Learning

REGRESSION MODEL

Raihana Fawaz (1103210102)

Import Library & Memuat Dataset

Import Library

```
▶ import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns
```

mengimport library yang digunakan

Lima Baris Pertama:

	Year	x1	x2	x3	x4	x5	x6	x7	\
0	1996	36.30414	12.66950	-41.26114	26.18958	-34.68851	-17.96802	12.54885	
1	1998	46.55790	22.35268	43.99892	5.47901	-9.96934	-8.28115	12.09128	
2	2000	40.20224	-3.51438	-5.17134	10.76605	-54.87086	31.64542	16.58214	
3	2007	50.54002	56.40521	12.87812	-6.06801	-35.50214	-20.55589	-8.54206	
4	1987	38.12363	0.13522	28.79411	-0.83252	-19.87921	-5.27092	-6.14879	

	x8	x9	...	x81	x82	x83	x84	x85	\
0	-1.00218	13.37747	...	13.48140	-36.96056	-35.33045	-91.40722	10.67250	
1	1.16916	5.14764	...	-1.47176	-0.02595	29.89710	65.84740	-3.20349	
2	0.16686	-4.23031	...	39.30145	-144.51193	84.67974	89.93842	0.96124	
3	-0.29037	9.85124	...	-4.68934	-58.75207	30.72153	54.11406	-3.36331	
4	0.12231	-16.71940	...	11.87244	-105.49613	-69.71921	1.61423	-3.50668	

	x86	x87	x88	x89	x90
0	-2.48525	2.30379	9.63628	65.20612	3.61463
1	8.00103	-5.33596	2.02504	53.91366	7.45981
2	337.52631	-9.75929	1.36992	199.09467	20.49871
3	-10.62442	36.59316	4.92336	-90.34671	1.42783
4	48.20428	54.15357	-12.27764	-124.89918	11.81774

5 baris pertama dataset

Tipe Data dan Duplikat



Tipe Data Kolom:

Year int64

x1 float64

x2 float64

x3 float64

x4 float64

...

x86 float64

x87 float64

x88 float64

x89 float64

x90 float64

Length: 91, dtype: object

Membersihkan baris yang mempunyai duplikat

```
▶ # Melihat baris yang memiliki duplikat  
duplicate_rows = dataset[dataset.duplicated()]  
print("\nBaris yang memiliki duplikat:", duplicate_rows.shape[0])
```



Baris yang memiliki duplikat: 4

```
[ ] # Menghapus baris yang memiliki duplikat  
dataset = dataset.drop_duplicates()  
print("\nJumlah baris setelah menghapus duplikat:", dataset.shape[0])
```



Tipe Data dan Duplikat



Tipe Data Kolom:

Year int64

x1 float64

x2 float64

x3 float64

x4 float64

...

x86 float64

x87 float64

x88 float64

x89 float64

x90 float64

Length: 91, dtype: object

Membersihkan baris yang mempunyai duplikat

```
[ ] # Melihat baris yang memiliki duplikat  
duplicate_rows = dataset[dataset.duplicated()  
print("\nBaris yang memiliki duplikat:", duplicate_rows.shape[0])
```



Baris yang memiliki duplikat: 4

```
[ ] # Menghapus baris yang memiliki duplikat  
dataset = dataset.drop_duplicates()  
print("\nJumlah baris setelah menghapus duplikat:", dataset.shape[0])
```



Jumlah baris setelah menghapus duplikat: 14996

Tipe Data dan Duplikat

```
▶ # Menghitung jumlah nilai yang hilang (missing values) di setiap kolom  
missing_values = dataset.isnull().sum()  
  
# Menampilkan hanya kolom-kolom yang memiliki nilai hilang  
missing_values = missing_values[missing_values > 0]  
  
# Output hasilnya  
missing_values
```

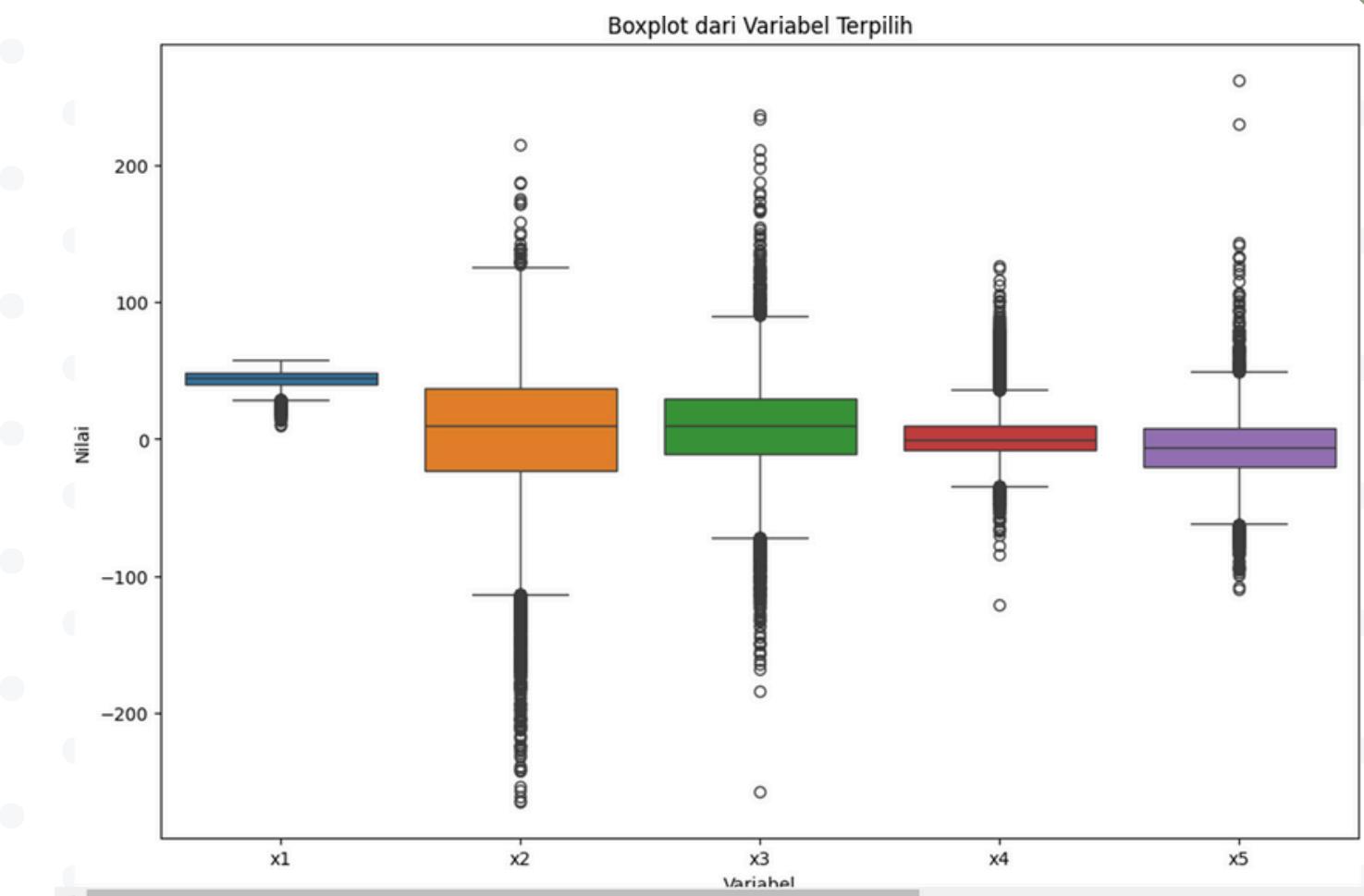


0

dtype: int64

Data Visualization

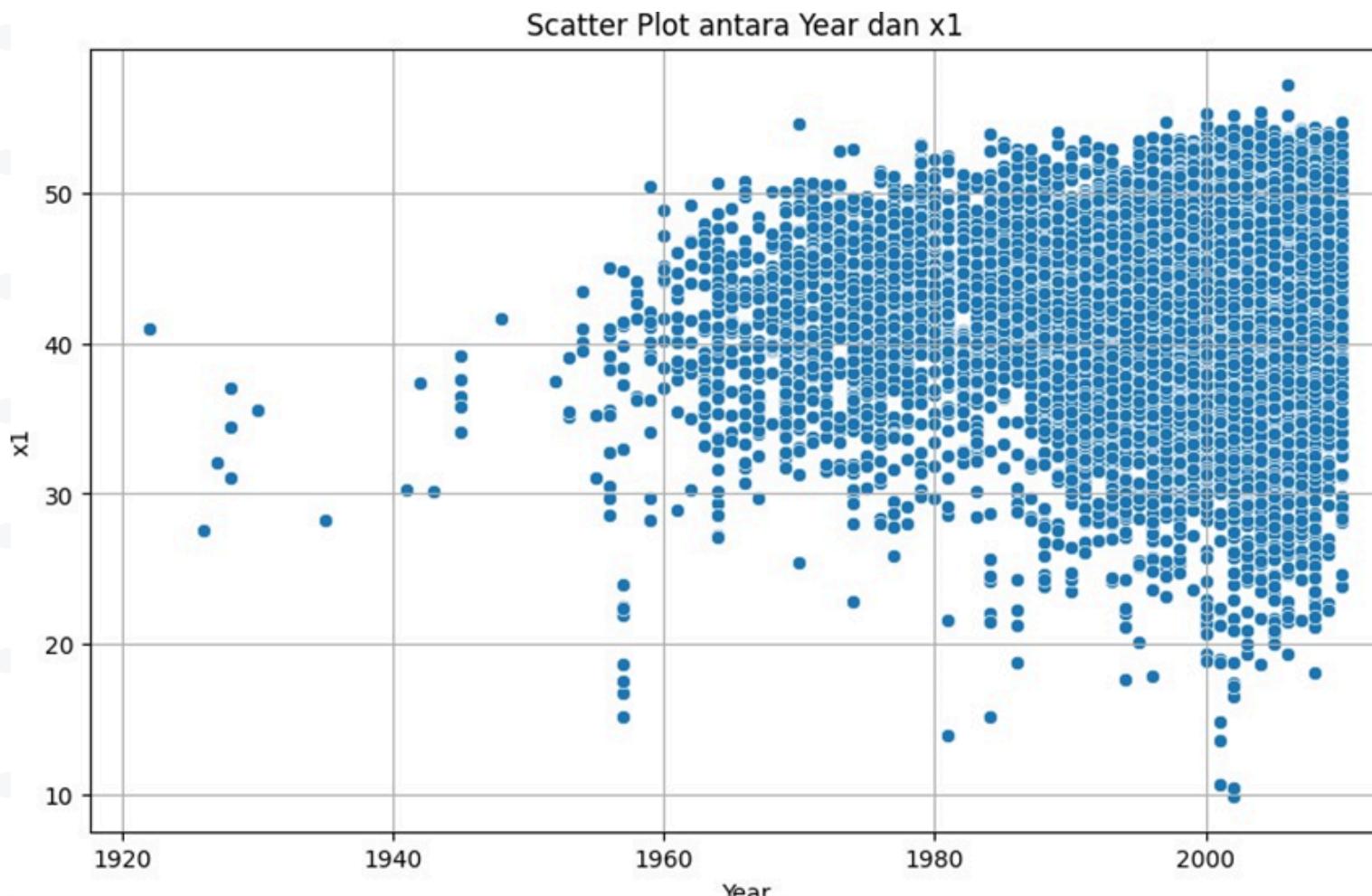
```
[ ] # Memilih beberapa variabel untuk visualisasi boxplot  
selected_variables = ['x1', 'x2', 'x3', 'x4', 'x5']  
  
# Membuat boxplot untuk variabel-variabel yang dipilih  
plt.figure(figsize=(12, 8)) # Menentukan ukuran plot  
sns.boxplot(data=dataset[selected_variables]) # Membuat boxplot untuk variabel terpilih  
plt.title('Boxplot dari Variabel Terpilih') # Memberikan judul pada plot  
plt.xlabel('Variabel') # Memberikan label pada sumbu X  
plt.ylabel('Nilai') # Memberikan label pada sumbu Y  
plt.show() # Menampilkan plot
```



Insight: Variabel seperti x3 memiliki rentang nilai yang lebih luas dibandingkan dengan x1 dan x4, menunjukkan bahwa datanya lebih bervariasi.

Scatter Plot

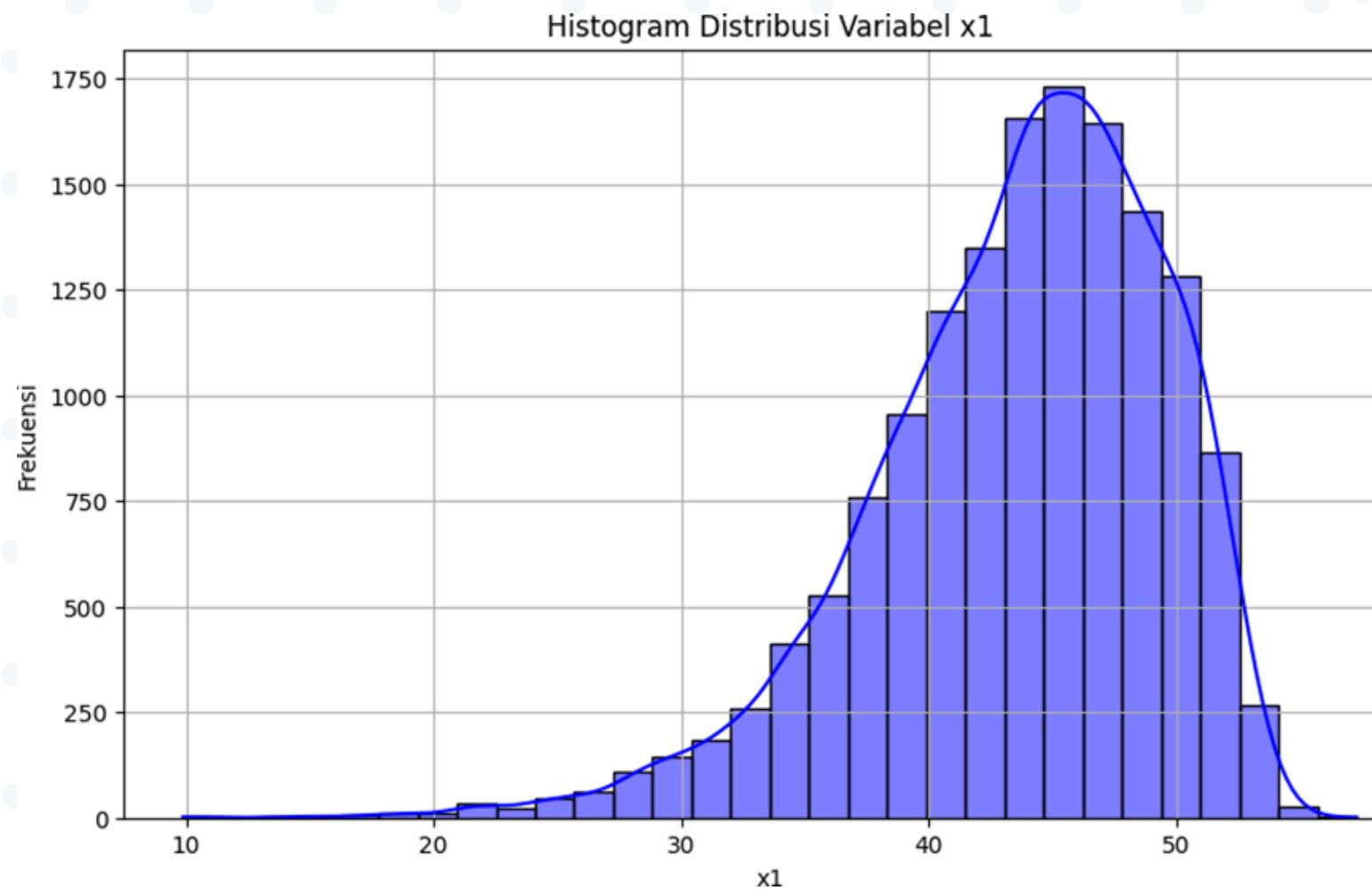
```
▶ # Membuat scatter plot untuk melihat hubungan antara 'x1' dan 'Year'  
plt.figure(figsize=(10, 6)) # Menentukan ukuran plot  
sns.scatterplot(x=dataset['Year'], y=dataset['x1']) # Membuat scatter plot dengan 'Year' sebagai sumbu X dan 'x1' sebagai sumbu Y  
plt.title('Scatter Plot antara Year dan x1') # Memberikan judul pada plot  
plt.xlabel('Year') # Memberikan label pada sumbu X  
plt.ylabel('x1') # Memberikan label pada sumbu Y  
plt.grid(True) # Menambahkan grid pada plot untuk memperjelas  
plt.show() # Menampilkan plot
```



Insight: Data mencakup rentangwaktu yang cukup panjang, daritahun awal (1922) hingga yanglebih baru, tetapi distribusinya tidak merata.

Histogram

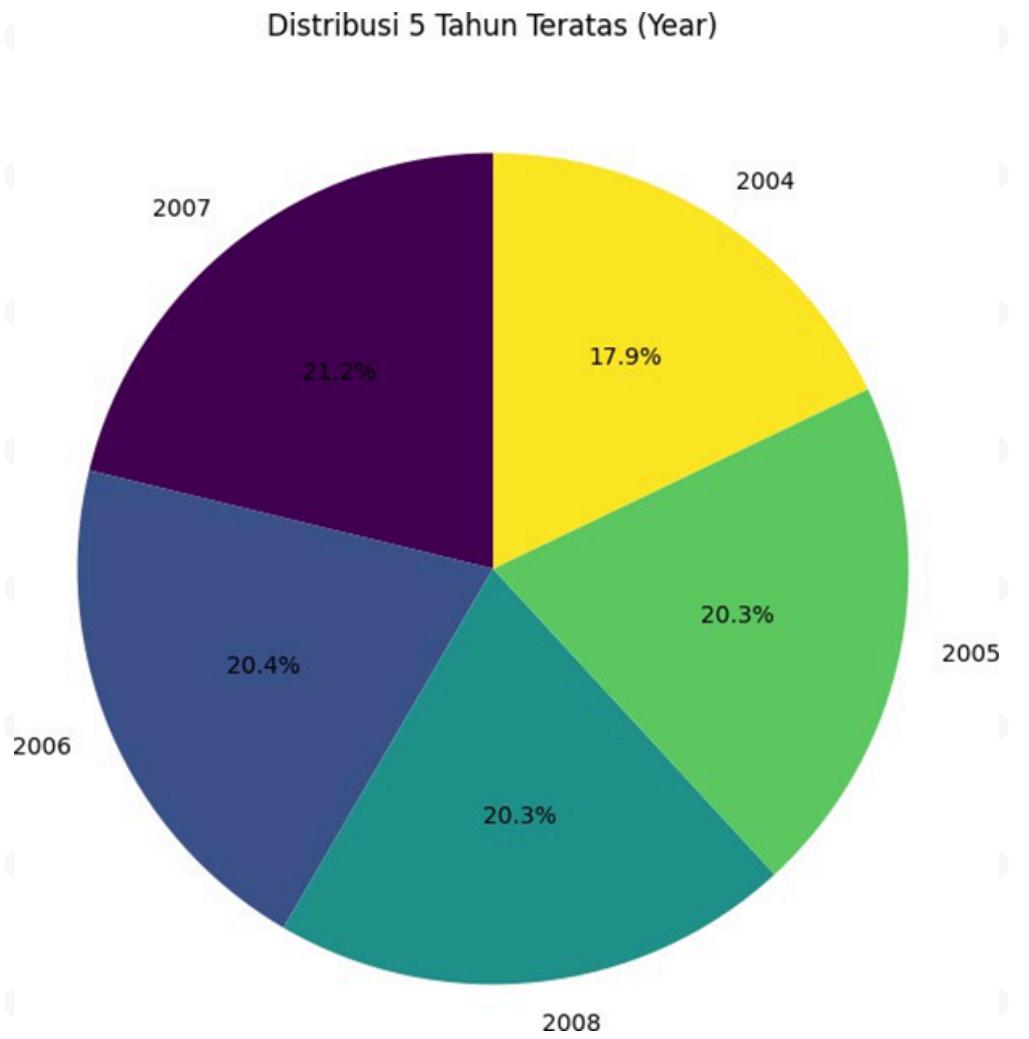
```
# Membuat histogram untuk melihat distribusi variabel 'x1'  
plt.figure(figsize=(10, 6)) # Menentukan ukuran plot  
sns.histplot(dataset['x1'], kde=True, bins=30, color='blue') # Membuat histogram dengan 30 bins dan menambahkan garis KDE  
plt.title('Histogram Distribusi Variabel x1') # Memberikan judul pada plot  
plt.xlabel('x1') # Memberikan label pada sumbu X  
plt.ylabel('Frekuensi') # Memberikan label pada sumbu Y  
plt.grid(True) # Menambahkan grid pada plot untuk memperjelas  
plt.show() # Menampilkan plot
```



Insight: Sebagian besar nilai x1 berada dalam rentang 40-50, menunjukkan bahwa data memiliki kecenderungan terpusat di sekitar rentang tersebut.

Pie Chart

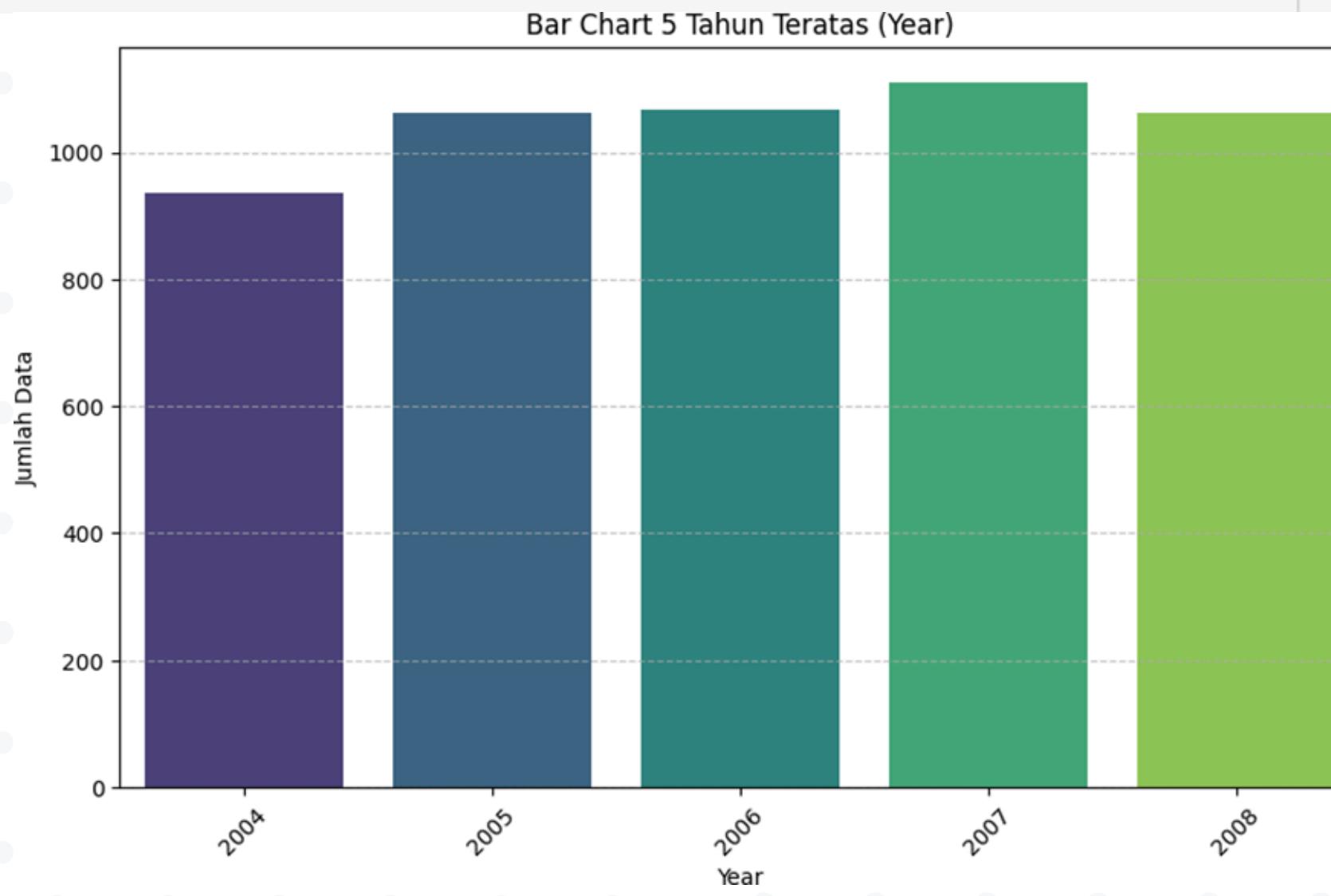
```
▶ # Membuat pie chart untuk 5 tahun dengan jumlah data terbanyak  
top_5_years = dataset['Year'].value_counts().head(5) # Mengambil 5 tahun teratas berdasarkan jumlah data  
  
plt.figure(figsize=(10, 8)) # Menentukan ukuran plot  
top_5_years.plot.pie(autopct='%1.1f%%', startangle=90, cmap='viridis') # Membuat pie chart  
plt.title('Distribusi 5 Tahun Teratas (Year)') # Memberikan judul pada chart  
plt.ylabel('') # Menghilangkan label sumbu Y untuk tampilan yang lebih bersih  
plt.show() # Menampilkan plot
```



Insight: Lima tahun teratas mencakup sebagian besar data, menunjukkan distribusi yang sangat terpusat pada periode tertentu.

Bar Chart

```
# Membuat bar chart untuk 5 tahun dengan jumlah data terbanyak
plt.figure(figsize=(10, 6)) # Menentukan ukuran plot
sns.barplot(x=top_5_years.index, y=top_5_years.values, palette='viridis') # Membuat bar chart
plt.title('Bar Chart 5 Tahun Teratas (Year)') # Memberikan judul pada chart
plt.xlabel('Year') # Memberikan label pada sumbu X
plt.ylabel('Jumlah Data') # Memberikan label pada sumbu Y
plt.xticks(rotation=45) # Memiringkan label sumbu X untuk keterbacaan
plt.grid(axis='y', linestyle='--', alpha=0.7) # Menambahkan grid horizontal untuk memperjelas
plt.show() # Menampilkan plot
```



Logistics Regression

Pipeline & Hyper Tuning

Data splitting

```
[ ] # Memisahkan dataset menjadi fitur (X) dan target (y)
X = dataset.drop(columns=['Year'])
y = dataset['Year']

[ ] # Membagi dataset menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Polynomial dan Hyperparameter Tuning

```
[ ] # Membuat pipeline untuk Polynomial Regression
pipeline = Pipeline([
    ('scaler', StandardScaler()), # Menstandarisasi fitur
    ('poly_features', PolynomialFeatures()), # Membuat fitur polinomial
    ('linear_regression', LinearRegression()) # Model regresi linier
])

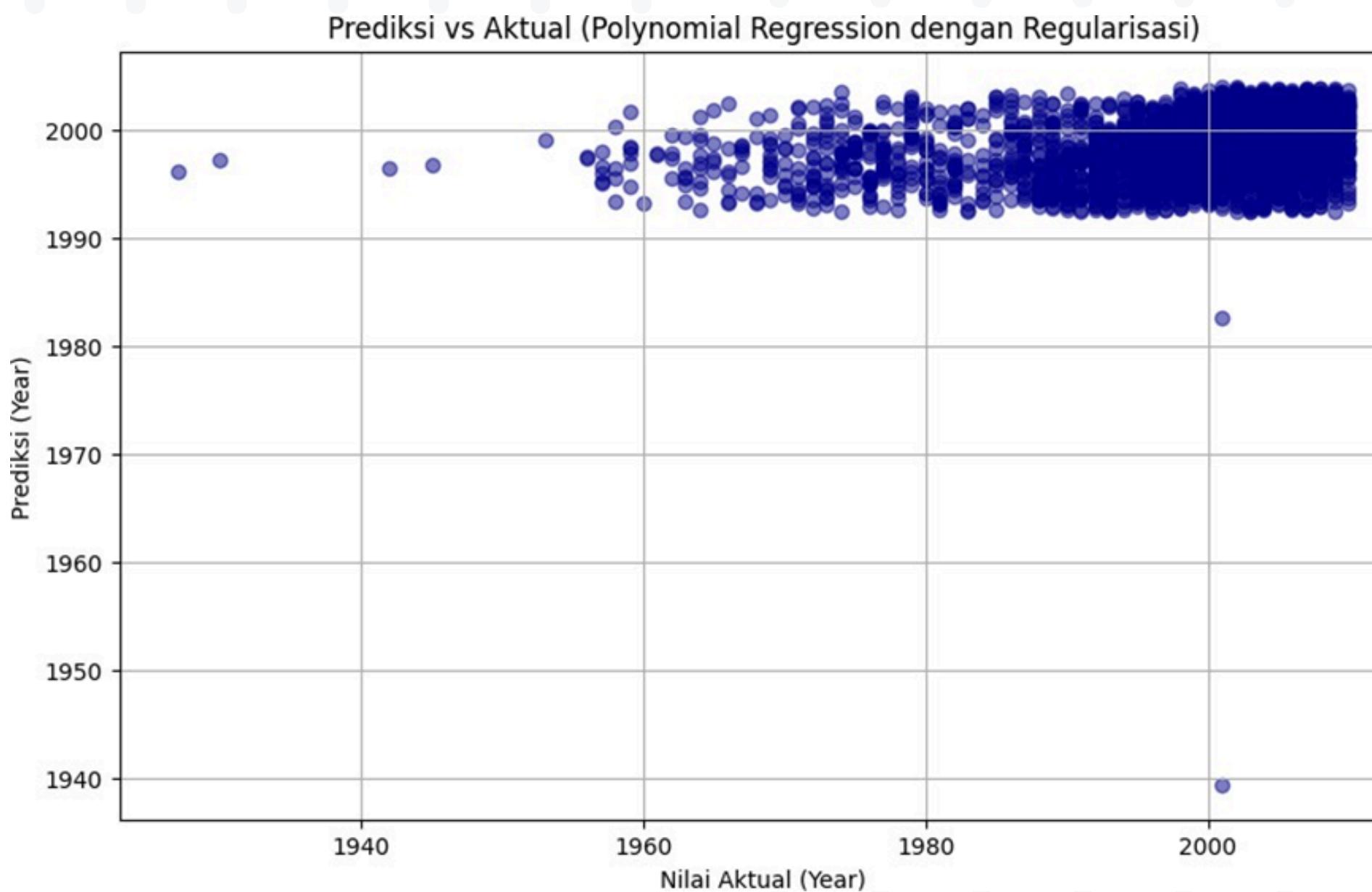
▶ # Membatasi pencarian hyperparameter untuk mempercepat proses
param_grid_limited = {
    'poly_features_degree': [2, 3], # Membatasi derajat polinomial ke 2 dan 3
    'linear_regression_fit_intercept': [True] # Menggunakan hanya True untuk fit_intercept
}
```

Model Evaluation

Model evaluation

```
[ ] # Menyimpan model terbaik dan mengevaluasinya  
best_model_limited = grid_search_limited.best_estimator_  
y_pred_limited = best_model_limited.predict(X_test)  
  
▶ # Menghitung metrik evaluasi  
mse_limited = mean_squared_error(y_test, y_pred_limited)  
r2_limited = r2_score(y_test, y_pred_limited)  
  
[ ] # Menampilkan hasil  
print("Hyperparameter Terbaik (Versi Terbatas):", grid_search_limited.best_params_)  
print("Mean Squared Error (MSE):", mse_limited)  
print("R-squared (R2):", r2_limited)  
  
→ Hyperparameter Terbaik (Versi Terbatas): {'linear_regression__fit_intercept': True, 'poly_features__degree': 3}  
Mean Squared Error (MSE): 1646.980148399938  
R-squared (R2): 0.3322065891834256
```

Visualisasi



Parameter Terbaik:

Mean Squared Error (MSE): 106.36150445831304

R-squared (R²): 0.06861351762549017

Decision Tree

Importing library

```
▶ # Mengimpor pustaka yang diperlukan
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.pipeline import Pipeline
    from sklearn.model_selection import GridSearchCV
    from sklearn.metrics import mean_squared_error, r2_score
```

Data splitting

```
[ ] # Memisahkan dataset menjadi fitur (X) dan target (y)
    X = dataset.drop(columns=['Year'])
    y = dataset['Year']

▶ # Membagi dataset menjadi data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Pipeline

Membuat pipeline

```
[ ] # Membuat pipeline untuk Decision Tree Regressor
pipeline_dt = Pipeline([
    ('decision_tree', DecisionTreeRegressor(random_state=42)) # Model Decision Tree
])
```

```
[ ] # Membuat parameter grid untuk Decision Tree Regressor
param_grid_dt = {
    'decision_tree__max_depth': [3, 5, 10, None], # Kedalaman maksimum pohon
    'decision_tree__min_samples_split': [2, 5, 10], # Minimum sampel untuk split
    'decision_tree__min_samples_leaf': [1, 2, 4], # Minimum sampel pada daun pohon
    'decision_tree__max_features': [None, 'sqrt', 'log2'] # Jumlah fitur untuk dipertimbangkan saat membelah
}
# Removed the duplicate entries and the incorrect indentation
```

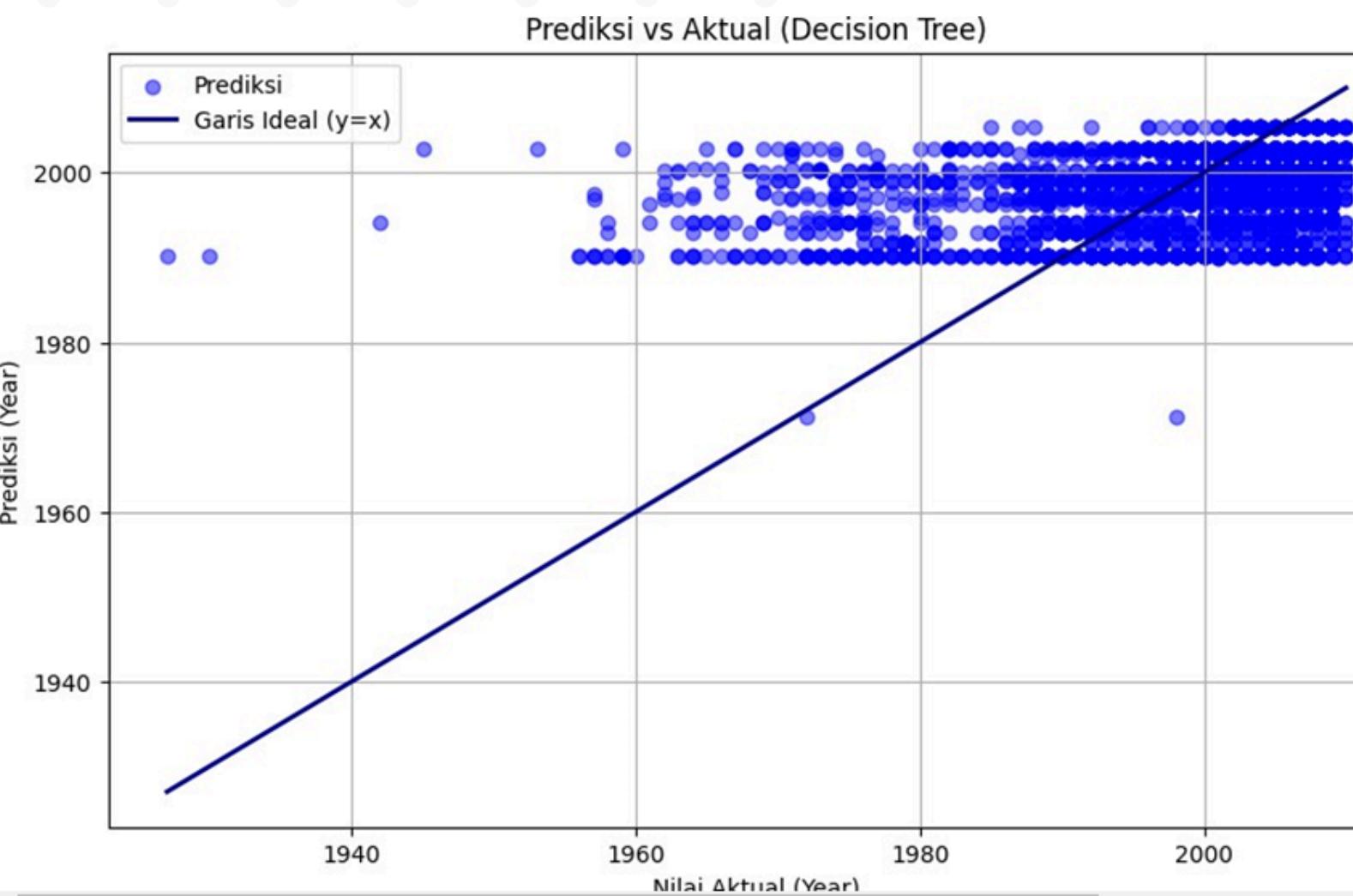
```
▶ # GridSearchCV untuk hyperparameter tuning
grid_search_dt = GridSearchCV(estimator=pipeline_dt, param_grid=param_grid_dt, cv=5, scoring='r2', n_jobs=-1)
grid_search_dt.fit(X_train, y_train)
```

Hypermeter Tuning

Hyperparameter tuning

```
▶ # Menampilkan hasil hyperparameter terbaik dan evaluasi model  
print("Hyperparameter Terbaik:", grid_search_dt.best_params_)  
print("Mean Squared Error (MSE):", mse_best)  
print("R-squared (R2):", r2_best)  
  
→ Hyperparameter Terbaik: {'decision_tree__max_depth': 5, 'decision_tree__criterion': 'gini'}  
Mean Squared Error (MSE): 98.77572611468689  
R-squared (R2): 0.13504066571375206
```

Visualisasi



- Pola yang lebih simetris di sepanjang garis diagonal menunjukkan bahwa model mampu menangkap pola dalam data.
- Jika ada penyebaran asimetris, misalnya banyak prediksi yang terlalu rendah atau terlalu tinggi, model mungkin mengalami bias.

K-Nearest Neighbors

Import Library & Data Splitting

Import library

```
▶ # Mengimpor pustaka yang diperlukan
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.neighbors import KNeighborsRegressor
    from sklearn.pipeline import Pipeline
    from sklearn.model_selection import GridSearchCV
    from sklearn.metrics import mean_squared_error, r2_score
```

+ Code

+ Text

Data splitting

```
[ ] # Memisahkan dataset menjadi fitur (X) dan target (y)
    X = dataset.drop(columns=['Year'])
    y = dataset['Year']
```

```
[ ] # Membagi dataset menjadi data latih dan data uji
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Pipeline

Membuat pipeline

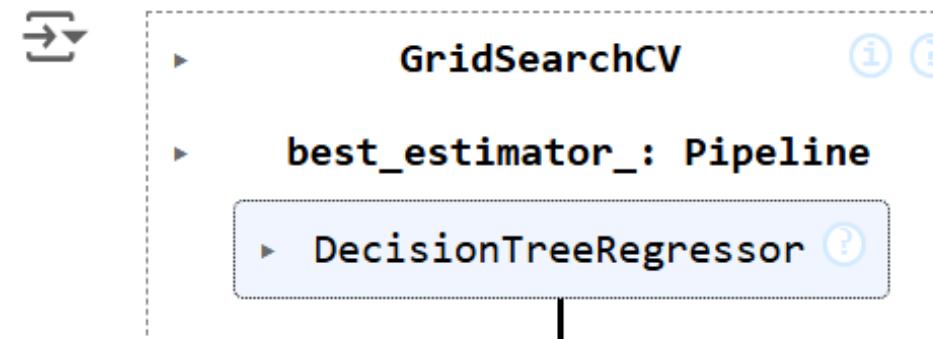
```
▶ # Membuat pipeline dengan k-Nearest Neighbors Regressor
pipeline = Pipeline([
    ('scaler', StandardScaler()), # Menstandarisasi fitur
    ('knn', KNeighborsRegressor()) # Model k-NN Regression
])
```

Hyperparameter Tuning

Hyperparameter tuning

```
▶ # Menentukan parameter untuk hyperparameter tuning
param_grid = {
    'knn__n_neighbors': [3, 5, 7, 10], # Jumlah tetangga
    'knn__weights': ['uniform', 'distance'], # Bobot sampel: uniform atau berbasis jarak
    'knn__p': [1, 2] # Parameter untuk jarak Minkowski: 1 untuk jarak Manhattan, 2 untuk jarak Euclidean
}
```

```
[ ] # GridSearchCV untuk hyperparameter tuning
grid_search_dt = GridSearchCV(estimator=pipeline_dt, param_grid=param_grid_dt, cv=5, scoring='r2', n_jobs=-1)
grid_search_dt.fit(X_train, y_train)
```



Model Evaluation

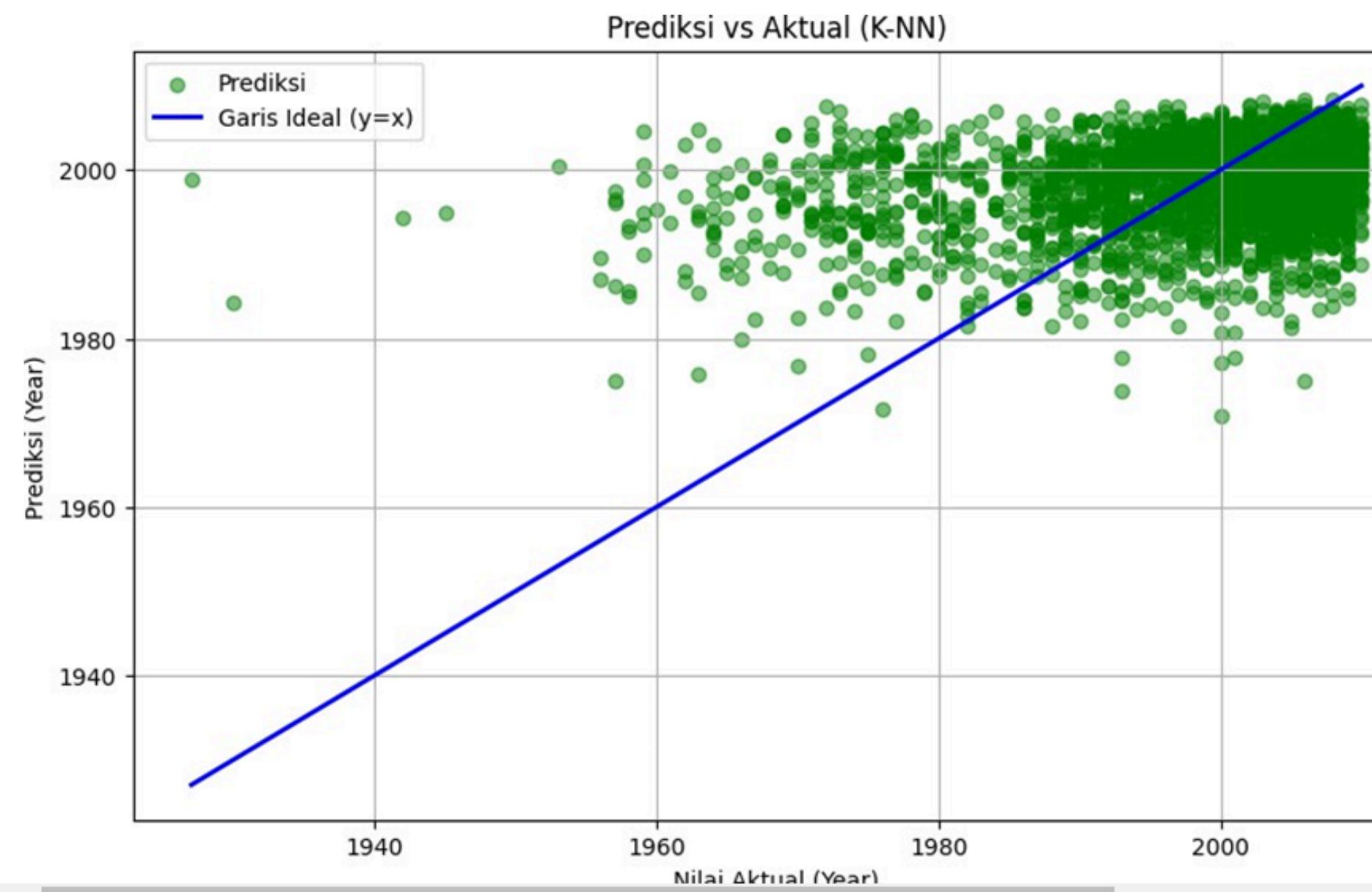


```
# Menampilkan hasil hyperparameter terbaik dan evaluasi model
print("Hyperparameter Terbaik:", gr Loading... dt.best_params_)
print("Mean Squared Error (MSE):", mse_best)
print("R-squared (R2):", r2_best)
```



```
Hyperparameter Terbaik: {'decision_tree__max_depth': 5, 'decision_tree__max_fe
Mean Squared Error (MSE): 98.77572611468689
R-squared (R2): 0.13504066571375206
```

Visualisasi



Jika titik hijau dalam scatter plot mendekati garis diagonal biru, model K-NN berhasil memprediksi nilai Year dengan akurasi yang baik.

XGBoost Regression

Importing library

```
▶ # Mengimpor pustaka yang diperlukan
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBRegressor
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
```

Data splitting

```
[ ] # Memisahkan dataset menjadi fitur (X) dan target (y)
X = dataset.drop(columns=['Year'])
y = dataset['Year']
```

```
[ ] # Membagi dataset menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Membuat pipeline

```
[ ] # Membuat model XGBoost Regressor
xgb_model = XGBRegressor(objective='reg:squarederror', random_state=42)
```

Hyperparameter tuning

```
▶ # Membuat parameter grid untuk RandomizedSearchCV
param_grid_xgb = {
    'n_estimators': [50, 100, 200], # Jumlah pohon
    'max_depth': [3, 5, 7], # Kedalaman maksimum pohon
    'learning_rate': [0.01, 0.1, 0.2], # Tingkat pembelajaran
    'subsample': [0.6, 0.8, 1.0], # Rasio sampel
    'colsample_bytree': [0.6, 0.8, 1.0] # Rasio fitur
}

[ ] # RandomizedSearchCV untuk hyperparameter tuning
from sklearn.model_selection import RandomizedSearchCV, train_test_split

randomized_search_xgb = RandomizedSearchCV(
    estimator=xgb_model,
    param_distributions=param_grid_xgb,
    n_iter=20, # Hanya mencoba 20 kombinasi
    cv=3, # Menggunakan 3-fold cross-validation
    scoring='r2',
    n_jobs=-1,
    random_state=42
```

Model Evaluation

Model Evaluation

```
[ ] # Model terbaik dari hasil RandomizedSearch
best_xgb_model = randomized_search_xgb.best_estimator_

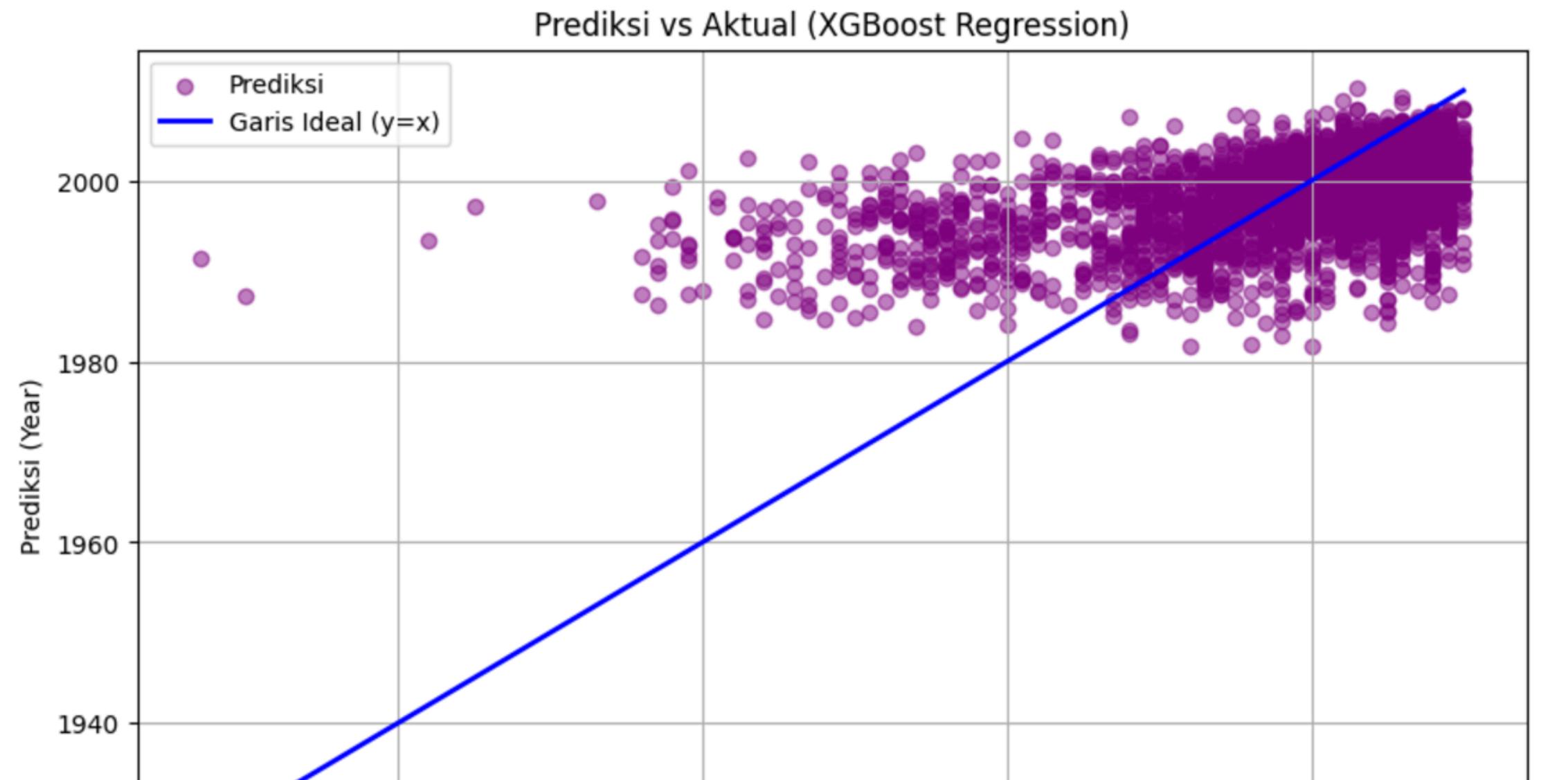
[ ] # Membuat prediksi menggunakan model terbaik
y_pred_xgb = best_xgb_model.predict(X_test)

[ ] # Evaluasi model XGBoost
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
r2_xgb = r2_score(y_test, y_pred_xgb)

# Menampilkan hasil hyperparameter terbaik dan evaluasi model
print("Hyperparameter Terbaik:", randomized_search_xgb.best_params_)
print("Mean Squared Error (MSE):", mse_xgb)
print("R-squared (R2):", r2_xgb)

→ Hyperparameter Terbaik: {'subsample': 1.0, 'n_estimators': 100, 'max_depth': 3, 'learning_rate': 0.1, 'colsample_bytree': 0.8}
Mean Squared Error (MSE): 87.57277686654031
R-squared (R2): 0.2331426739692688
```

Visualisasi



Sebagian besar titik prediksi (titik ungu) tersebar di atas garis diagonal biru (garis ideal $y = x$), yang menunjukkan bahwa model cenderung melebih-lebihkan nilai prediksi dibandingkan dengan nilai aktual Year.