Universitas Telkom

UTS MACHINE LEARNING

Regression Model

Oleh :

Ferdinant Hutajulu

1103213120

Universitas Telkom

```python
import pandas as pd

# Load the uploaded dataset
file_path = '/content/drive/MyDrive/dataset diperkecil.csv'
dataset = pd.read_csv(file_path)

# Display the first few rows to understand its structure
dataset.head()
```

	x2	x4	x7	x8	x37	x41	x47	x64	x68	x1
0	48.73215	70.32679	-24.83777	8.76630	26.84939	202.18689	-12.19034	12.59163	-88.37882	2001
1	50.95714	55.81851	-18.54940	-3.27872	28.70107	13.09302	-7.27994	1.75729	-12.14279	2001
2	48.24750	36.29772	-26.21683	5.05097	35.63919	-0.01744	-7.20736	-11.63176	-83.15795	2001
3	50.97020	67.09964	-16.81409	-12.48207	11.20673	28.68782	-15.55296	-4.05003	-47.75605	2001
4	50.54767	92.35066	-19.04928	20.67345	20.84154	51.41086	-9.11273	0.51657	-9.41185	2001

- Fungsi: Code disamping digunakan untuk mengimpor pustaka pandas untuk membaca file CSV dan menampilkan beberapa baris pertama dataset.
- Tujuan: Membaca dan memahami dataset awal.

Universitas Telkom

```
[6] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Code diatas digunakan untuk menghubungkan Google Drive dengan Google Colab agar dataset yang telah di upload ke Google Drive bisa terbaca di Google Colab.

Universitas Telkom

```python
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Basic Info and Statistics
eda_report = dataset.describe().transpose()

# Check for missing values
missing_values = dataset.isnull().sum()

# Display EDA statistics and missing values
eda_report, missing_values
```

Selanjutya code diatas berguan untuk memberikan statistik deskriptif seperti rata-rata, standar deviasi, dan quartile untuk setiap kolom numerik juga berguna sebagai penghitung nilai kosong yang ada disetiap kolom.

```
(        count         mean         std         min          25%          50%
 x2   515128.0    43.386224    6.067920     1.74900    39.953392    44.257065
 x4   515128.0     8.658907   35.270848  -301.00506   -11.463037    10.476855
 x7   515128.0    -9.521505   12.858281   -81.79429   -18.441175   -11.187815
 x8   515128.0    -2.391041   14.572861  -188.21400   -10.780360    -2.047015
 x37  515128.0    72.654053  107.918153 -1711.48400    14.478230    56.222440
 x41  515128.0    23.094897  205.753394 -7882.82324   -69.684438    21.138595
 x47  515128.0     6.337656   54.977994 -1329.95974   -17.315575     3.957220
 x64  515128.0    -0.475633   37.678071  -600.09076   -18.716963    -3.296740
 x68  515128.0     3.156558   99.927003 -1199.00442   -46.434375     2.260245
 x1   515128.0  1998.396272   10.931651  1922.00000  1994.000000  2002.000000

              75%         max
 x2     47.833525    61.97014
 x4     29.766687   322.85143
 x7     -2.387157   166.23689
 x8      6.508798   172.40268
 x37   116.851642  2496.12262
 x41   115.104765  8360.14557
 x47    28.017140  1198.62677
 x64    14.852250   812.42425
 x68    51.596412  2055.03948
 x1   2006.000000  2011.00000  ,
 x2     0
 x4     0
 x7     0
 x8     0
 x37    0
 x41    0
 x47    0
 x64    0
 x68    0
 x1     0
 dtype: int64)
```

Universitas Telkom

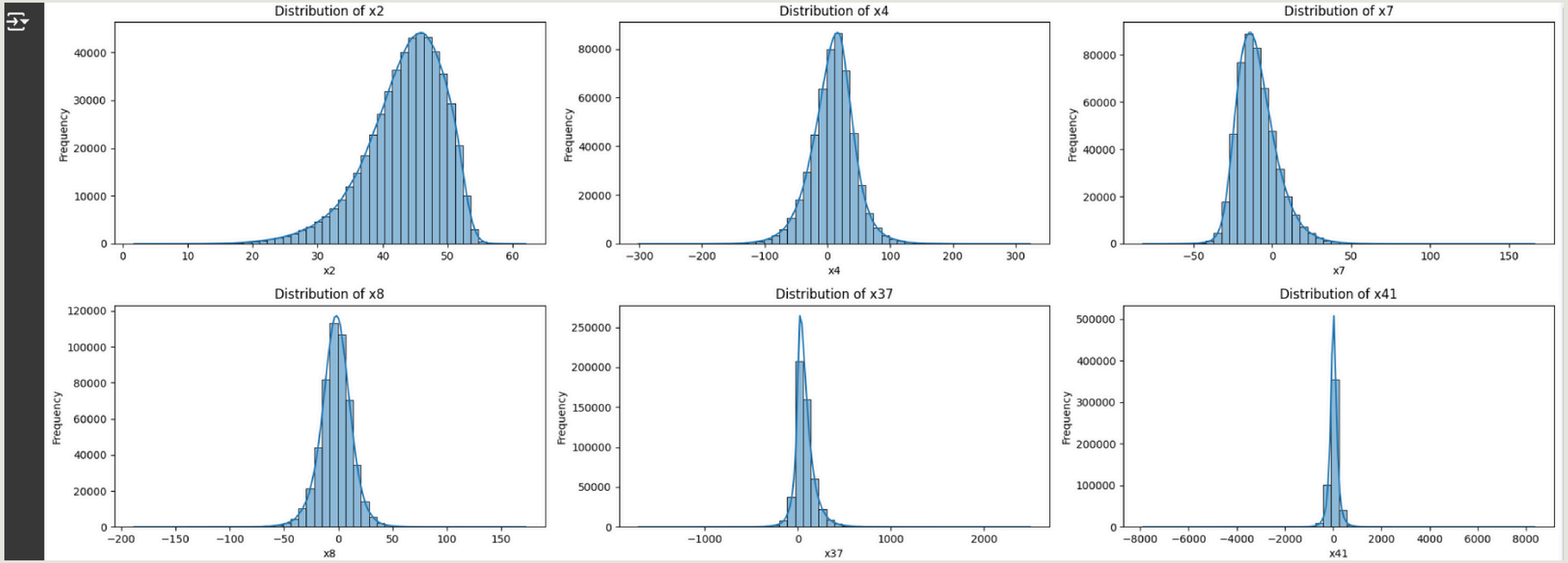
```
[8] # Visualize the distributions of numerical features
    numerical_cols = dataset.columns

    plt.figure(figsize=(20, 15))
    for idx, col in enumerate(numerical_cols, 1):
        plt.subplot(4, 3, idx)
        sns.histplot(dataset[col], kde=True, bins=50)
        plt.title(f'Distribution of {col}')
        plt.xlabel(col)
        plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
```

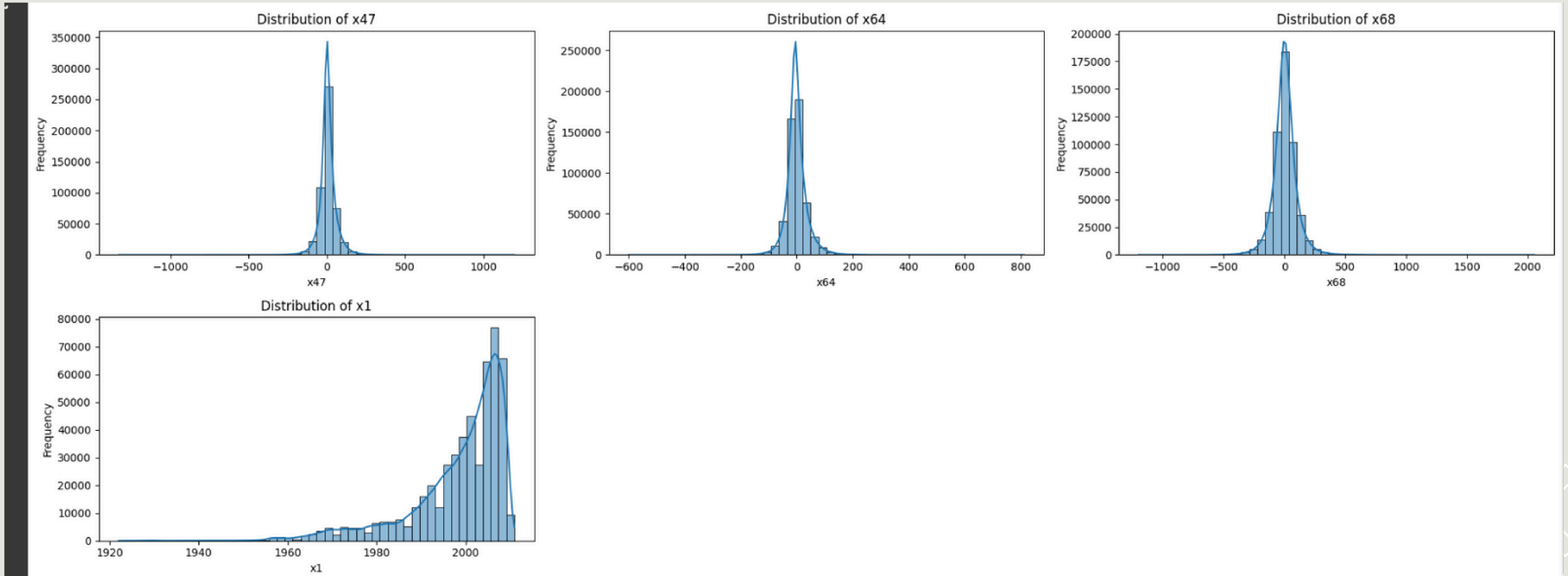
Fungsi:

- Membuat histogram untuk setiap kolom numerik di dataset menggunakan Seaborn.
- Menambahkan garis distribusi kernel (KDE) untuk melihat pola distribusi data.

Tujuan: Memahami distribusi data di setiap kolom dan mengidentifikasi pola seperti simetri, pencilan, atau bias.

Universitas Telkom



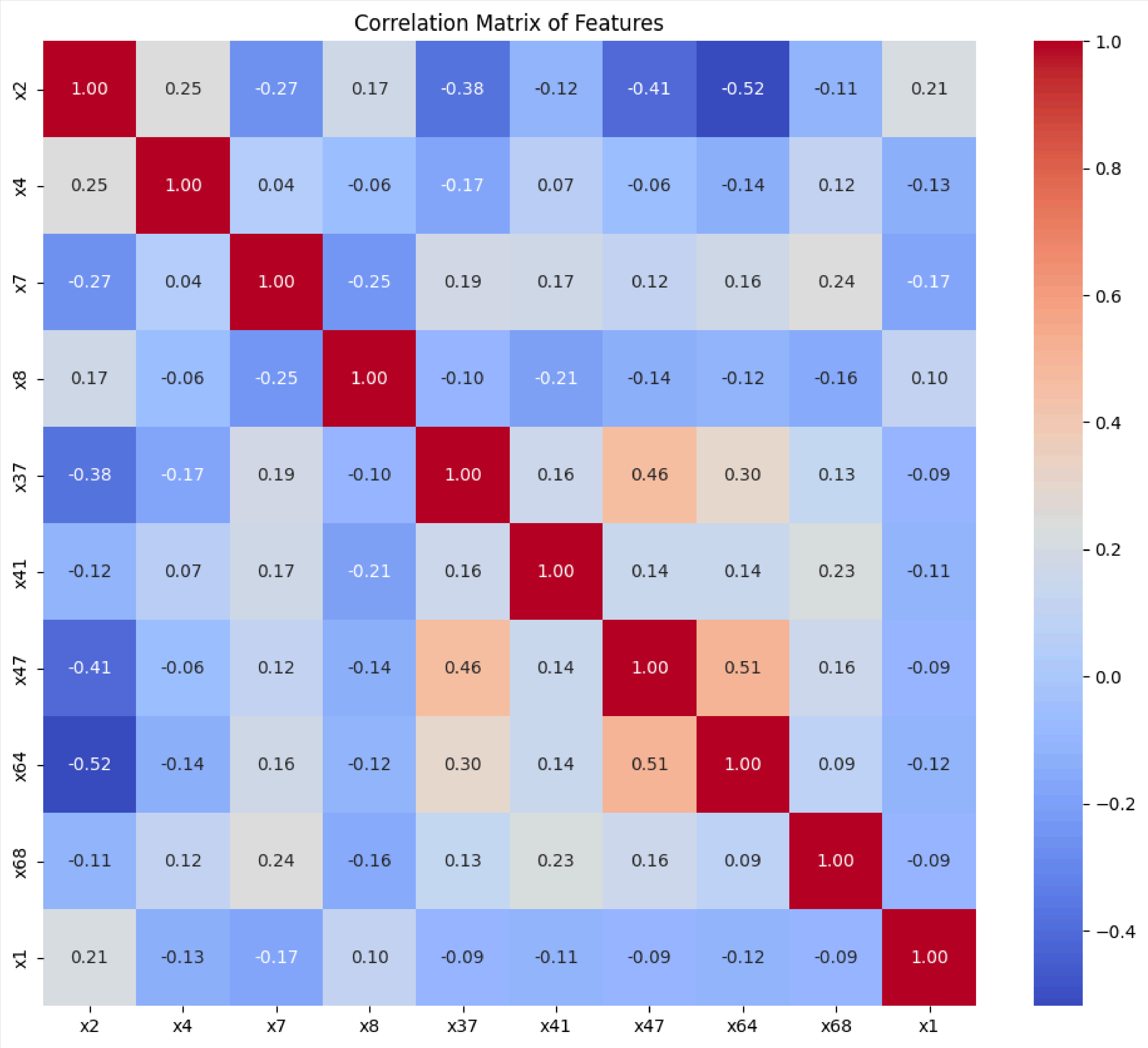


Universitas Telkom

```python
# Compute the correlation matrix
correlation_matrix = dataset.corr()

# Visualize the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix of Features")
plt.show()
```

- Fungsi: Menghitung dan memvisualisasikan hubungan antar fitur menggunakan matriks korelasi.
- Tujuan: Mengidentifikasi kolerasi yang kuat antar variabel.

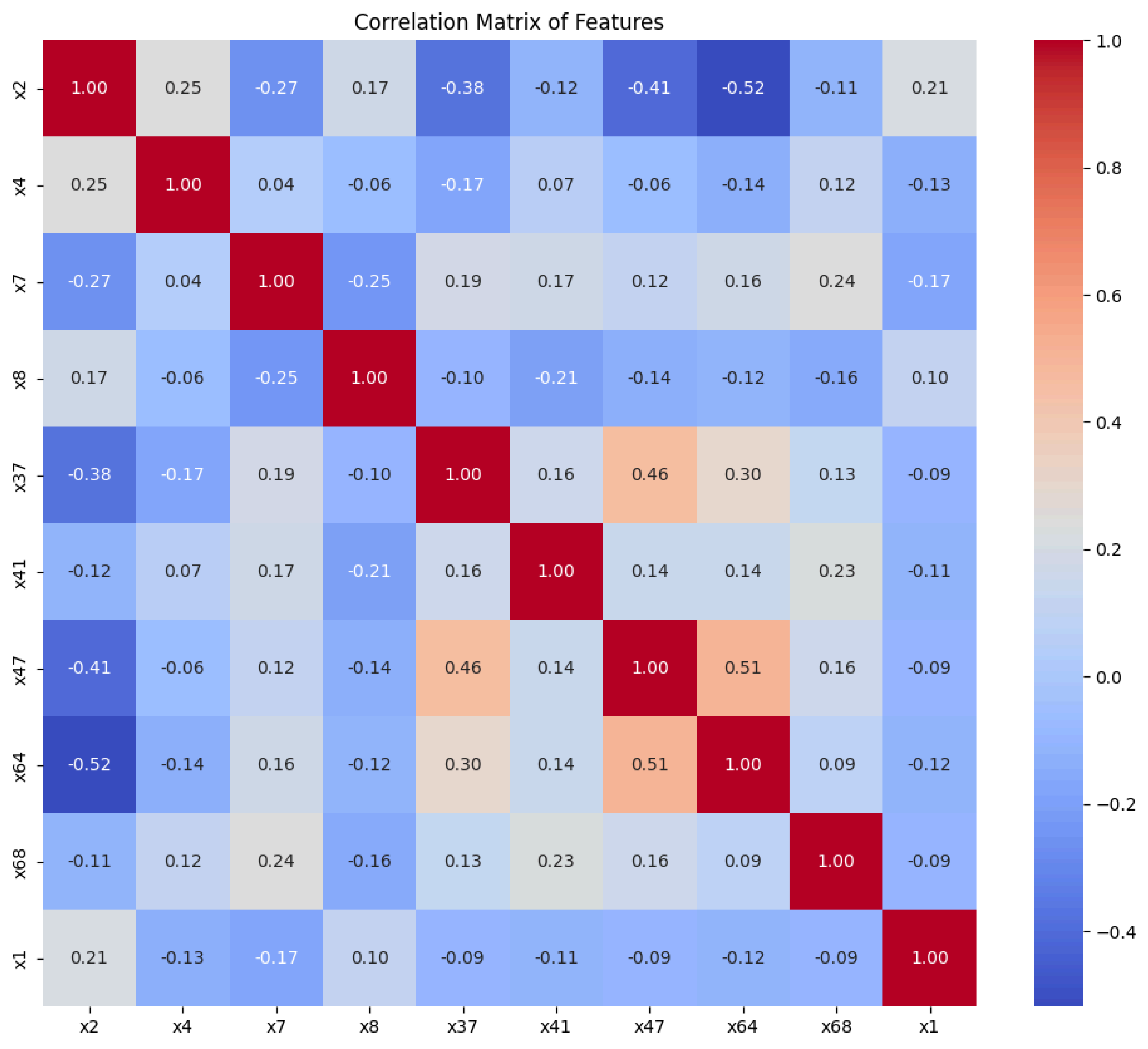
Universitas Telkom



```python
# Retry visualizing the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix of Features")
plt.show()
```

Code disamping hanya berguna untuk mengulang hasil dari code yang sebelumnya.

Universitas Telkom



Universitas Telkom

```python
# Import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load dataset
file_path = '/content/drive/MyDrive/dataset diperkecil.csv'  # Replace with your actual file path
dataset = pd.read_csv(file_path)

# Split dataset into features (X) and target (y)
X = dataset.drop(columns=['x1'])  # Assuming 'x1' is the target column
y = dataset['x1']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define pipelines for each model
pipelines = {
    "Polynomial Regression": Pipeline([
        ('poly', PolynomialFeatures()),  # Generates polynomial and interaction features
        ('scaler', StandardScaler()),  # Standardizes the features
        ('model', LinearRegression())  # Linear regression model
    ]),
    "Decision Tree": Pipeline([
        ('model', DecisionTreeRegressor(random_state=42))  # Decision tree model
```

```python
    ]),
    "k-NN": Pipeline([
        ('scaler', StandardScaler()),  # Standardizes the features
        ('model', KNeighborsRegressor())  # k-NN regressor
    ])
}

# Define parameter grids for hyperparameter tuning
param_grids = {
    "Polynomial Regression": {'poly__degree': [1, 2, 3]},  # Tuning polynomial degree
    "Decision Tree": {'model__max_depth': [5, 10, 20],  # Tuning max depth of the tree
                      'model__min_samples_split': [2, 5, 10]},  # Tuning min samples to split
    "k-NN": {'model__n_neighbors': [3, 5, 10]}  # Tuning number of neighbors
}

# Perform grid search for each model
evaluation_results = []
for model_name, pipeline in pipelines.items():
    # Grid Search                                 (variable) model_name: str
    grid_search = GridSearchCV(pipeline, param_grids[model_name], cv=3, scoring='neg_mean_squared_error', n_jobs=-1)
    grid_search.fit(X_train, y_train)

    # Best model
    best_model = grid_search.best_estimator_

    # Predictions
    y_pred = best_model.predict(X_test)

    # Evaluation metrics
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
```

Universitas Telkom

```python
    r2 = r2_score(y_test, y_pred)

    # Store results
    evaluation_results.append({
        "Model": model_name,
        "Best Parameters": grid_search.best_params_,
        "MSE": mse,
        "MAE": mae,
        "R²": r2
    })

# Convert results to a DataFrame and print
evaluation_df = pd.DataFrame(evaluation_results)
print("Model Evaluation Results:")
print(evaluation_df)
```

Keseluruhan fungsi code yang ada dihalaman sebelunya dan yang ada disamping adalah :

- **Melakukan pencarian parameter terbaik untuk setiap model.**
- **Menyimpan hasil pencarian (parameter terbaik dan skor terbaik).**
- **Membandingkan skor MSE terbaik melalui grafik batang horizontal.**
- **Mencetak parameter terbaik untuk dokumentasi.**

Universitas Telkom

```
Model Evaluation Results:
                  Model                                     Best Parameters
0  Polynomial Regression                              {'poly__degree': 3}
1          Decision Tree  {'model__max_depth': 10, 'model__min_samples_s...
2                   k-NN                       {'model__n_neighbors': 10}

          MSE       MAE        R²
0  101.742672  7.432665  0.139487
1  102.455091  7.379568  0.133461
2  106.451915  7.558325  0.099657
```

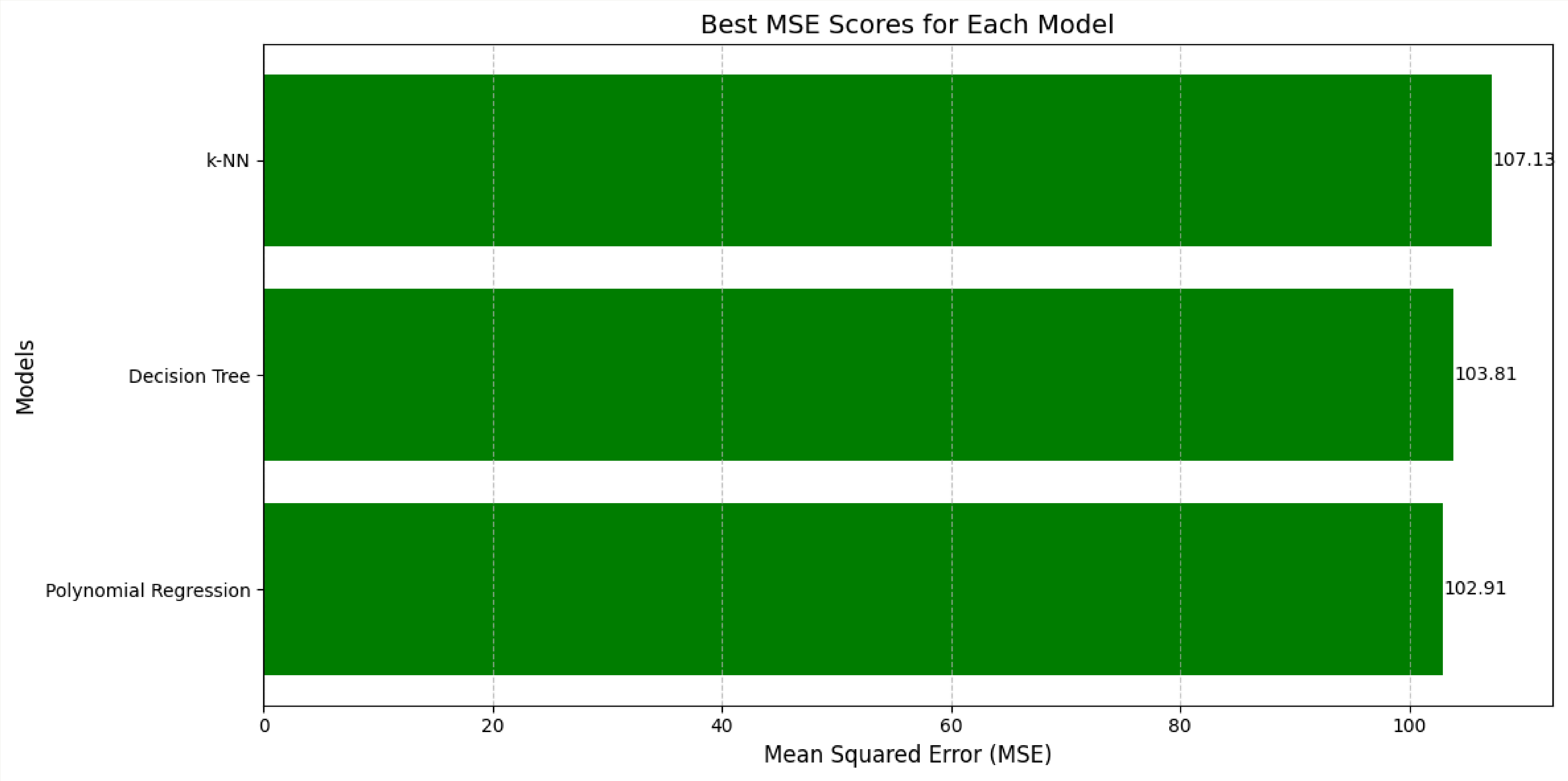
Universitas
Telkom

```python
import matplotlib.pyplot as plt
import numpy as np

# Perform grid search for the remaining models
results = {}
for model_name, pipeline in pipelines.items():
    grid_search = GridSearchCV(pipeline, param_grids[model_name], cv=3, scoring='neg_mean_squared_error', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    results[model_name] = {
        'best_params': grid_search.best_params_,
        'best_score': -grid_search.best_score_  # Convert negative MSE to positive for interpretability
    }

# Prepare data for visualization
models = list(results.keys())
scores = [result['best_score'] for result in results.values()]
best_params = [str(result['best_params']) for result in results.values()]

# Plot results
plt.figure(figsize=(12, 6))
plt.barh(models, scores, color='green')
for i, v in enumerate(scores):
    plt.text(v + 0.1, i, f"{v:.2f}", va='center', fontsize=10)

plt.title('Best MSE Scores for Each Model', fontsize=14)
plt.xlabel('Mean Squared Error (MSE)', fontsize=12)
plt.ylabel('Models', fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```python
# Print Best Parameters for Reference
print("Best Parameters for Each Model:")
for model, params in zip(models, best_params):
    print(f"{model}: {params}")
```

Code ini memiliki fungsi sebagai berikut :

- Melakukan pencarian hyperparameter terbaik untuk setiap model menggunakan grid search.
- Mengonversi hasil pencarian ke bentuk DataFrame untuk kemudahan analisis.
- Membuat visualisasi skor MSE terbaik untuk setiap model dalam bentuk grafik batang.
- Menampilkan hasil tuning parameter terbaik dalam format teks untuk referensi mendetail.



Best MSE Scores for Each Model
k-NN
Decision Tree
Polynomial Regression
107.13
103.81
102.91
0
20
40
60
80
100
Mean Squared Error (MSE)
Models

Universitas Telkom

```python
import matplotlib.pyplot as plt

# Perform grid search for the remaining models
results = {}
for model_name, pipeline in pipelines.items():
    grid_search = GridSearchCV(pipeline, param_grids[model_name], cv=3, scoring='neg_mean_squared_error', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    results[model_name] = {
        'best_params': grid_search.best_params_,
        'best_score': -grid_search.best_score_  # Convert negative MSE to positive for interpretability
    }

# Convert results to DataFrame
results_df = pd.DataFrame(results).transpose()

# Plotting MSE of models
plt.figure(figsize=(10, 6))
models = results_df.index
mse_values = results_df['best_score']

plt.bar(models, mse_values, color='grey')
plt.xlabel("Models")
plt.ylabel("Best Mean Squared Error (MSE)")
plt.title("Hyperparameter Tuning Results for Regression Models")
plt.xticks(rotation=45)
plt.tight_layout()

# Display the plot
plt.show()
```

```python
# Print detailed results
print("Hyperparameter Tuning Results:")
print(results_df)
```

Universitas
Telkom

```python
import matplotlib.pyplot as plt

# Redefine train-test split
X = dataset.drop(columns=['x1'])  # Assuming 'x1' is the target column
y = dataset['x1']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Perform grid search for the adjusted models
results = {}
for model_name, pipeline in pipelines.items():
    grid_search = GridSearchCV(pipeline, param_grids[model_name], cv=3, scoring='neg_mean_squared_error', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    results[model_name] = {
        'best_params': grid_search.best_params_,
        'best_score': -grid_search.best_score_  # Convert negative MSE to positive for interpretability
    }

# Convert results to a DataFrame
results_df = pd.DataFrame(results).transpose()

# Plot the best scores (MSE) for the models
plt.figure(figsize=(10, 6))
models = results_df.index
mse_values = results_df['best_score']

plt.bar(models, mse_values, color='grey')
plt.xlabel("Models")
plt.ylabel("Best Mean Squared Error (MSE)")
plt.title("Hyperparameter Tuning Results for Polynomial Regression, Decision Tree, and k-NN")
plt.xticks(rotation=45)
```

plt.xticks(rotation=45)

```python
plt.tight_layout()

# Display the plot
plt.show()

# Print detailed results in the console
print("Hyperparameter Tuning Results:")
print(results_df)
```

Universitas Telkom

```python
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np
import pandas as pd

# Helper function to calculate metrics
def evaluate_model(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    return mse, rmse, mae, r2

# Polynomial Regression
poly_pipeline = Pipeline([
    ('poly', PolynomialFeatures()),
    ('scaler', StandardScaler()),
    ('model', LinearRegression())
])
poly_param_grid = {
    'poly__degree': [2, 3, 4],
    'model__fit_intercept': [True, False]
}
poly_grid = GridSearchCV(poly_pipeline, poly_param_grid, cv=5, scoring='neg_mean_squared_error')
poly_grid.fit(X_train, y_train)
```

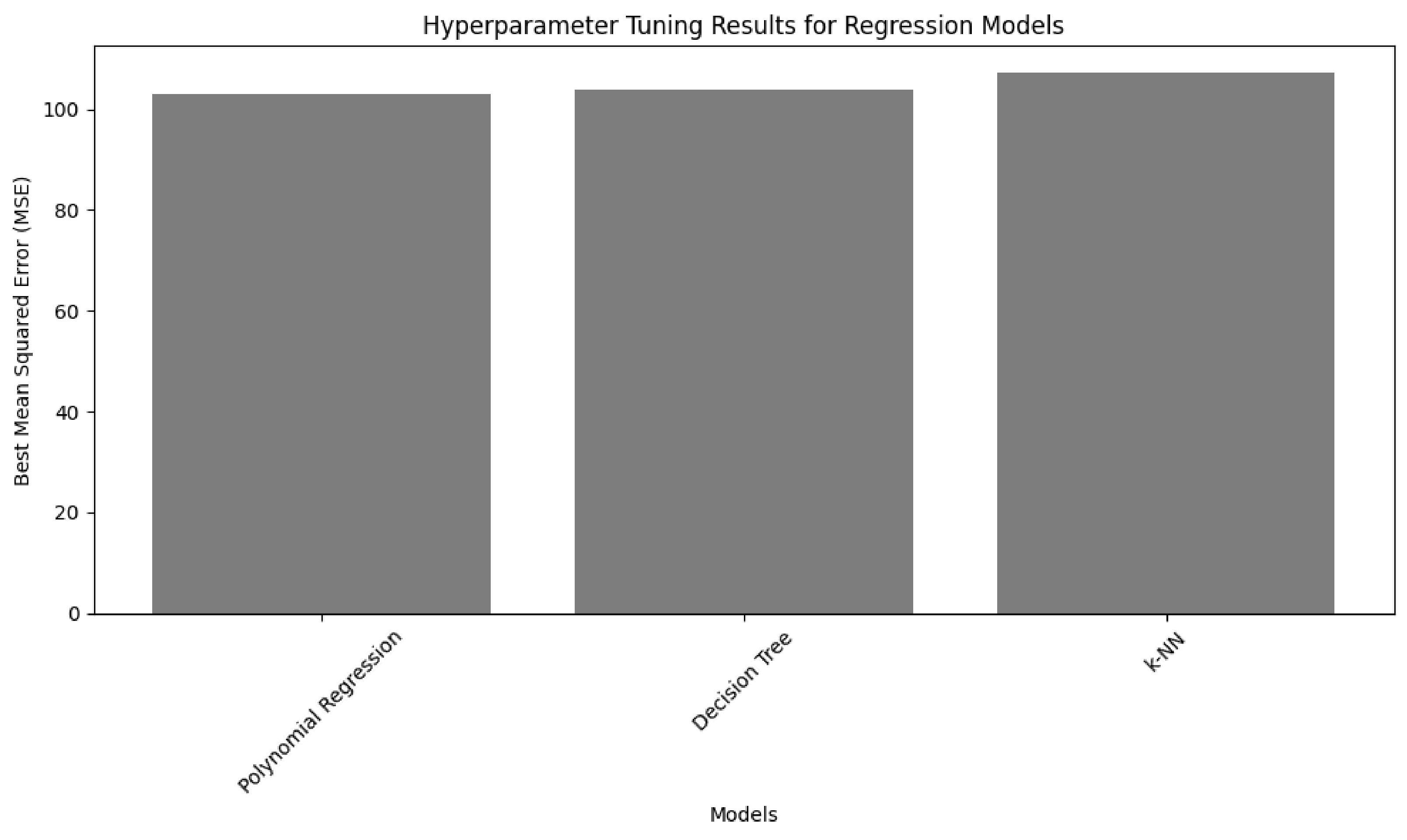
```python
poly_best_model = poly_grid.best_estimator_
poly_y_pred = poly_best_model.predict(X_test)
poly_mse, poly_rmse, poly_mae, poly_r2 = evaluate_model(y_test, poly_y_pred)
print("\nBest Polynomial Regression Model Metrics:")
print(f"MSE: {poly_mse:.4f}, RMSE: {poly_rmse:.4f}, MAE: {poly_mae:.4f}, R^2: {poly_r2:.4f}")

# Decision Tree
tree_param_grid = {
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
tree_grid = GridSearchCV(DecisionTreeRegressor(random_state=42), tree_param_grid, cv=5, scoring='neg_mean_squared_error')
tree_grid.fit(X_train, y_train)
tree_best_model = tree_grid.best_estimator_
tree_y_pred = tree_best_model.predict(X_test)
tree_mse, tree_rmse, tree_mae, tree_r2 = evaluate_model(y_test, tree_y_pred)
print("\nBest Decision Tree Model Metrics:")
print(f"MSE: {tree_mse:.4f}, RMSE: {tree_rmse:.4f}, MAE: {tree_mae:.4f}, R^2: {tree_r2:.4f}")

# k-Nearest Neighbors
knn_param_grid = {
    'n_neighbors': [3, 5, 10],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
knn_grid = GridSearchCV(KNeighborsRegressor(), knn_param_grid, cv=5, scoring='neg_mean_squared_error')
knn_grid.fit(X_train, y_train)
knn_best_model = knn_grid.best_estimator_
knn_y_pred = knn_best_model.predict(X_test)
knn_mse, knn_rmse, knn_mae, knn_r2 = evaluate_model(y_test, knn_y_pred)
```

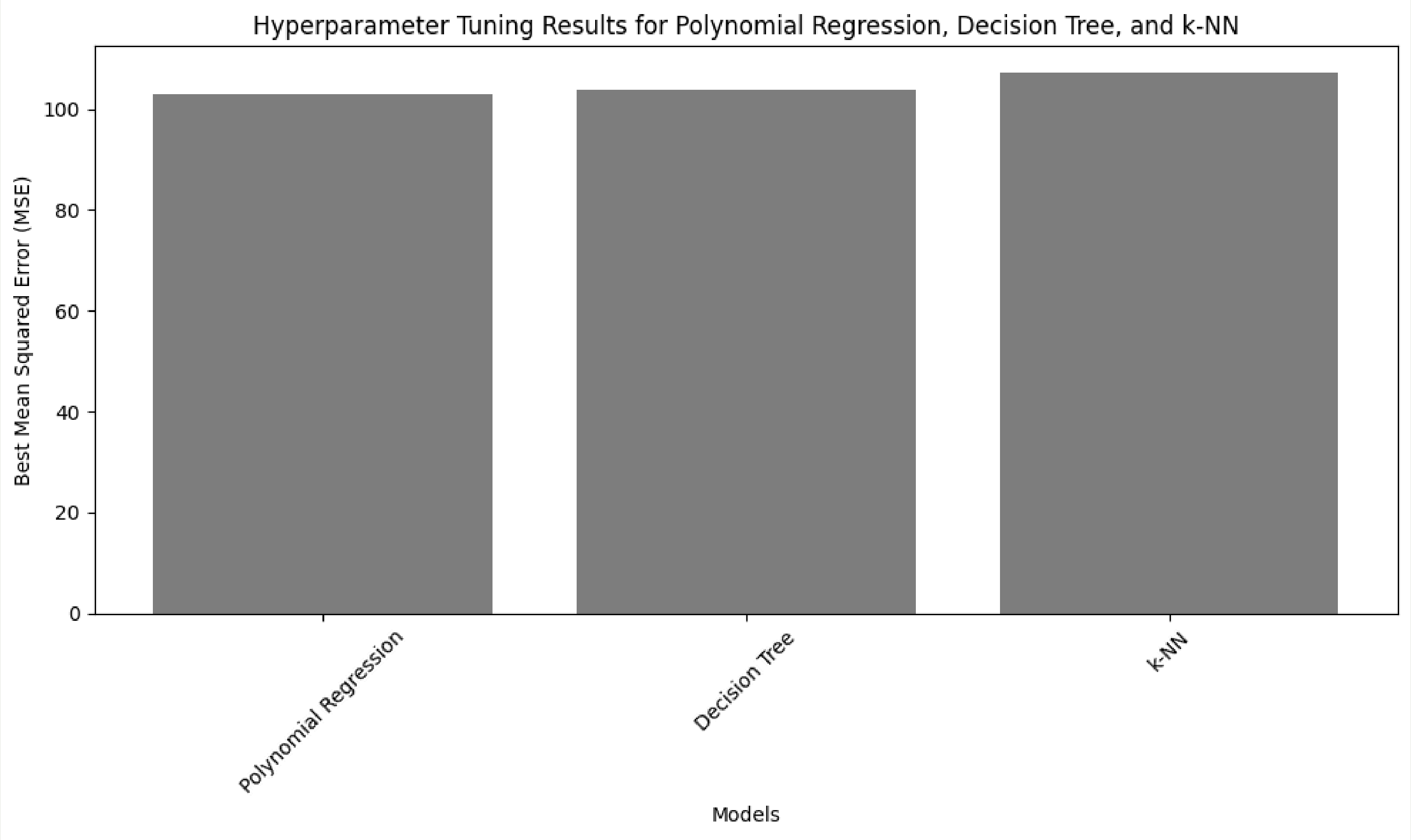
Universitas Telkom

```
print("\nBest k-Nearest Neighbors Model Metrics:")
print(f"MSE: {knn_mse:.4f}, RMSE: {knn_rmse:.4f}, MAE: {knn_mae:.4f}, R^2: {knn_r2:.4f}")

# XGBoost
xgb_param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
xgb_grid = GridSearchCV(XGBRegressor(random_state=42, eval_metric='rmse'), xgb_param_grid, cv=5, scoring='neg_mean_squared_error')
xgb_grid.fit(X_train, y_train)
xgb_best_model = xgb_grid.best_estimator_
xgb_y_pred = xgb_best_model.predict(X_test)
xgb_mse, xgb_rmse, xgb_mae, xgb_r2 = evaluate_model(y_test, xgb_y_pred)
print("\nBest XGBoost Model Metrics:")
print(f"MSE: {xgb_mse:.4f}, RMSE: {xgb_rmse:.4f}, MAE: {xgb_mae:.4f}, R^2: {xgb_r2:.4f}")
```

Keseluruhan fungsi code yang ada dihalaman sebelumnya adalah sebagai berikut :

- Membandingkan empat model regresi: Polynomial Regression, Decision Tree, k-NN, dan XGBoost.
- Melakukan tuning hyperparameter menggunakan GridSearchCV.
- Mengevaluasi performa model dengan metrik MSE, RMSE, MAE, dan R^2.
- Memberikan informasi tentang model terbaik untuk setiap metode.






```
Best Polynomial Regression Model Metrics:
MSE: 101.7427, RMSE: 10.0868, MAE: 7.4327, R^2: 0.1395

Best Decision Tree Model Metrics:
MSE: 102.1097, RMSE: 10.1049, MAE: 7.3740, R^2: 0.1364

Best k-Nearest Neighbors Model Metrics:
MSE: 111.5095, RMSE: 10.5598, MAE: 7.7954, R^2: 0.0569

Best XGBoost Model Metrics:
MSE: 97.9025, RMSE: 9.8946, MAE: 7.2119, R^2: 0.1720
```

Universitas Telkom

UTS MACHINE LEARNING

Classification Model

Oleh :

Ferdinant Hutajulu
1103213120

	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3	ZrO2	P2O5
0	FLQ-1-b	Body	0.62	0.38	19.61	71.99	4.84	0.31	0.07	1.18	630	10	70	10	430	0	40	80	90
1	FLQ-2-b	Body	0.57	0.47	21.19	70.09	4.98	0.49	0.09	1.12	380	20	80	40	430	-10	40	100	110
2	FLQ-3-b	Body	0.49	0.19	18.60	74.70	3.47	0.43	0.06	1.07	420	20	50	50	380	40	40	80	200
3	FLQ-4-b	Body	0.89	0.30	18.01	74.19	4.01	0.27	0.09	1.23	460	20	70	60	380	10	40	70	210
4	FLQ-5-b	Body	0.03	0.36	18.41	73.99	4.33	0.65	0.05	1.19	380	40	90	40	360	10	30	80	150

Beberapa baris pertama dari dataset yang diambil untuk memahami struktur data

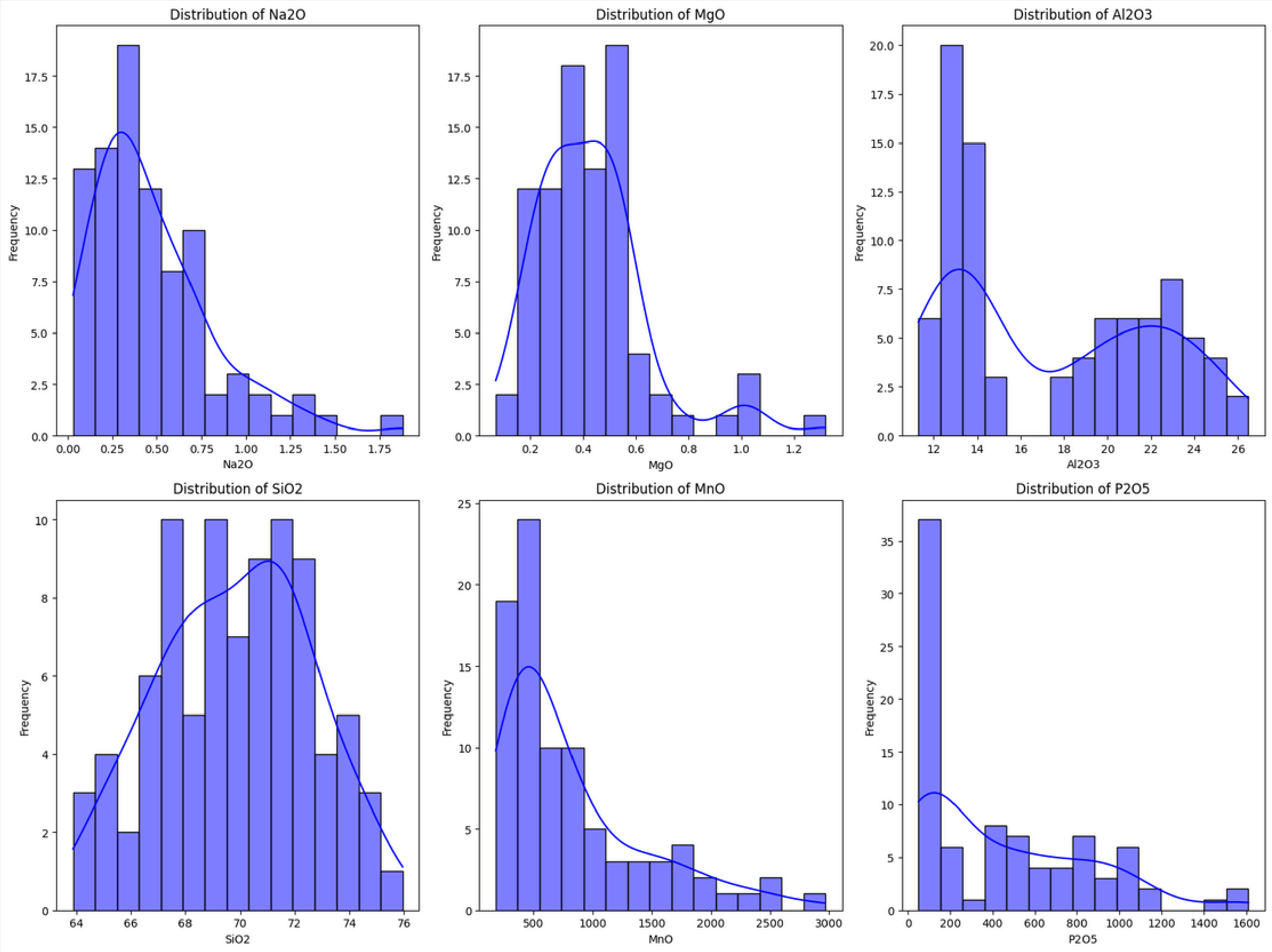
Universitas Telkom

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88 entries, 0 to 87
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Ceramic Name  88 non-null     object 
 1   Part          88 non-null     object 
 2   Na2O          88 non-null     float64
 3   MgO           88 non-null     float64
 4   Al2O3         88 non-null     float64
 5   SiO2          88 non-null     float64
 6   K2O           88 non-null     float64
 7   CaO           88 non-null     float64
 8   TiO2          88 non-null     float64
 9   Fe2O3         88 non-null     float64
 10  MnO           88 non-null     int64  
 11  CuO           88 non-null     int64  
 12  ZnO           88 non-null     int64  
 13  PbO2          88 non-null     int64  
 14  Rb2O          88 non-null     int64  
 15  SrO           88 non-null     int64  
 16  Y2O3          88 non-null     int64  
 17  ZrO2          88 non-null     int64  
 18  P2O5          88 non-null     int64  
dtypes: float64(8), int64(9), object(2)
memory usage: 13.2+ KB
```

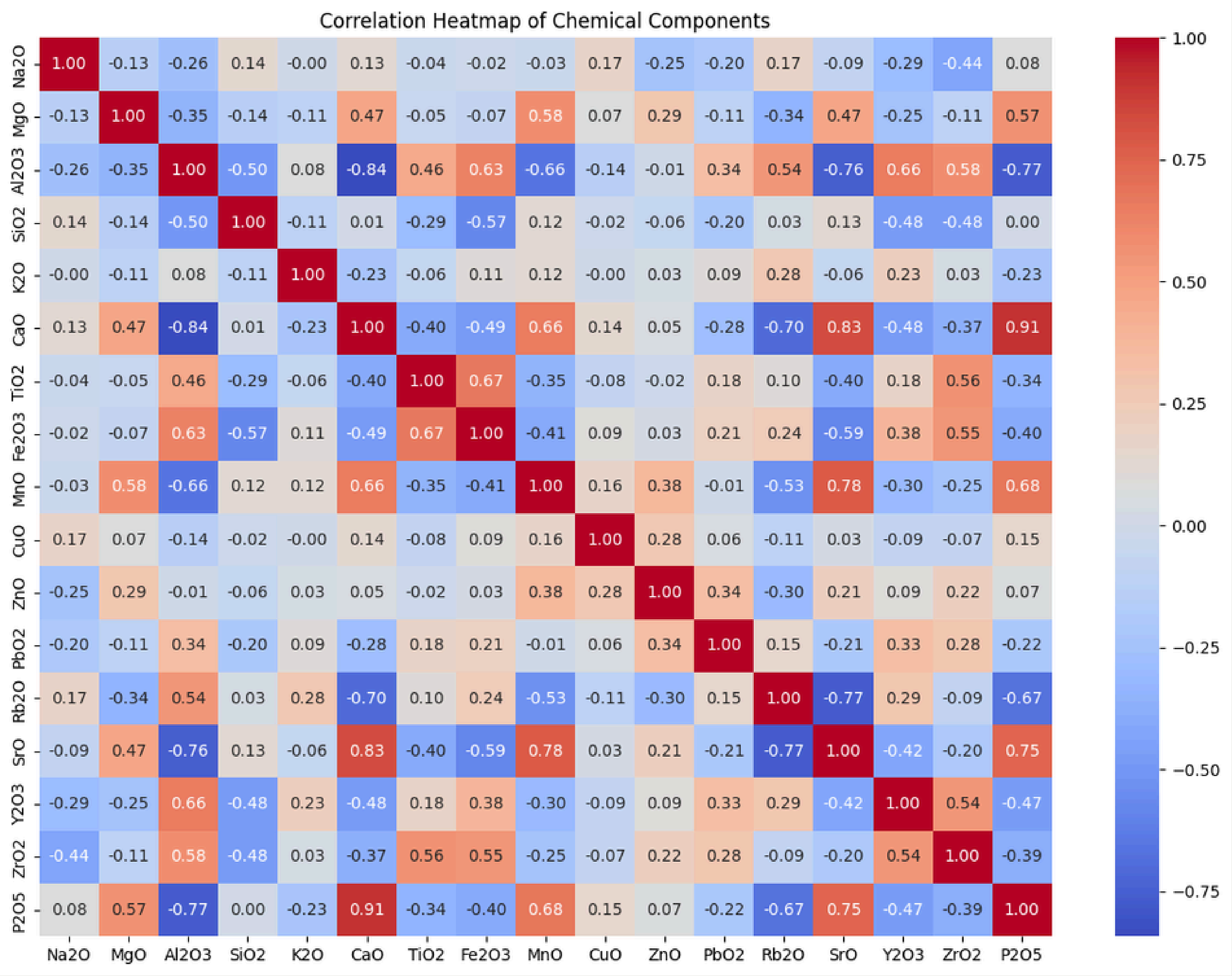
	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3	ZrO2	P2O5
count	88.000000	88.000000	88.000000	88.000000	88.000000	88.000000	88.00000	88.000000	88.000000	88.000000	88.000000	88.000000	88.000000	88.000000	88.000000	88.000000	88.000000
mean	0.471705	0.430114	17.460909	69.825114	4.978409	4.171818	0.10125	1.561591	818.750000	30.909091	95.340909	38.522727	310.454545	228.863636	42.954545	145.454545	440.909091
std	0.348779	0.215030	4.703422	2.754377	0.879467	4.305801	0.05343	0.604276	614.240607	19.096630	33.901441	26.589246	69.809414	256.216646	12.879556	60.074840	402.653944
min	0.030000	0.070000	11.300000	63.880000	2.730000	0.120000	0.04000	0.580000	180.000000	0.000000	20.000000	0.000000	180.000000	-10.000000	20.000000	50.000000	50.000000
25%	0.247500	0.270000	13.007500	67.737500	4.337500	0.180000	0.07000	1.097500	380.000000	20.000000	70.000000	20.000000	250.000000	10.000000	30.000000	100.000000	97.500000
50%	0.375000	0.405000	16.205000	69.990000	5.065000	2.690000	0.08000	1.510000	590.000000	30.000000	90.000000	30.000000	320.000000	75.000000	40.000000	140.000000	365.000000
75%	0.642500	0.530000	21.707500	71.840000	5.590000	7.912500	0.13000	1.925000	982.500000	40.000000	112.500000	60.000000	370.000000	482.500000	50.000000	170.000000	697.500000
max	1.880000	1.320000	26.480000	75.950000	6.740000	13.690000	0.29000	3.110000	2970.000000	80.000000	230.000000	100.000000	450.000000	780.000000	80.000000	390.000000	1610.000000

Output diatas berisi tentang:

- Informasi dataset: jumlah baris, kolom, tipe data, dan apakah ada nilai kosong
- Statistik deskriptif untuk kolom numerik



Gambar disamping menampilkan grafik plot distribusi data untuk beberapa kolom.



Menampilkan heatmap untuk melihat korelasi antar komponen kimia.

```
Best Parameters: {'C': 0.1, 'solver': 'lbfgs'}
Best Score: 0.9714285714285715
```

Menampilkan Best Parameters dan Best Score dari Logistic Regression.

```
Best Parameters: {'max_depth': 10, 'min_samples_split': 5}
Best Score: 1.0
```

Menampilkan Best Parameters dan Best Score dari Decision Tree.

```
Best Parameters: {'n_neighbors': 3, 'weights': 'uniform'}
Best Score: 0.9857142857142858
```

Menampilkan Best Parameters dan Best Score dari KNN.

Universitas Telkom

```
Best Parameters: {'learning_rate': 0.01, 'n_estimators': 50}
Best Score: 0.9857142857142858
              precision    recall  f1-score   support

        Body       1.00      1.00      1.00         8
       Glaze       1.00      1.00      1.00        10

    accuracy                           1.00        18
   macro avg       1.00      1.00      1.00        18
weighted avg       1.00      1.00      1.00        18
```

Laporan klasifikasi model XGBoost dengan GridSearchCV

Universitas Telkom

	precision	recall	f1-score	support
0	1.00	1.00	1.00	8
1	1.00	1.00	1.00	10
accuracy			1.00	18
macro avg	1.00	1.00	1.00	18
weighted avg	1.00	1.00	1.00	18

Laporan klasifikasi evaluasi model terbaik pada data uji.

Terima Kasih