TUGAS UTS

Muhamad Rizq Rihaz
(1103210192)

CLUSTERING MODEL

EXPLANATORY DATA ANALYSIS

CODE

```python
# Import library yang dibutuhkan
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Membaca dataset
url = '/content/heart_failure_clinical_records_dataset.csv'  # Ganti dengan path dataset yang benar
data = pd.read_csv(url)

# Menampilkan 5 baris pertama dari dataset
print(data.head())
```

HASIL

```
    age  anaemia  creatinine_phosphokinase  diabetes  ejection_fraction  \
0  75.0        0                       582         0                 20
1  55.0        0                      7861         0                 38
2  65.0        0                       146         0                 20
3  50.0        1                       111         0                 20
4  65.0        1                       160         1                 20

   high_blood_pressure  platelets  serum_creatinine  serum_sodium  sex  \
0                    1  265000.00               1.9           130    1
1                    0  263358.03               1.1           136    1
2                    0  162000.00               1.3           129    1
3                    0  210000.00               1.9           137    1
4                    0  327000.00               2.7           116    0

   smoking  time  DEATH_EVENT
0        0     4            1
1        0     6            1
2        1     7            1
3        0     7            1
4        0     8            1
```

EXPLANATORY DATA ANALYSIS

CODE

```
# Mengetahui informasi dasar tentang dataset
print(data.info())
```

HASIL

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64  
 2   creatinine_phosphokinase  299 non-null    int64  
 3   diabetes                  299 non-null    int64  
 4   ejection_fraction         299 non-null    int64  
 5   high_blood_pressure       299 non-null    int64  
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64  
 9   sex                       299 non-null    int64  
 10  smoking                   299 non-null    int64  
 11  time                      299 non-null    int64  
 12  DEATH_EVENT               299 non-null    int64  
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
None
```

EXPLANATORY DATA ANALYSIS

CODE

```python
# Deskripsi statistik dari dataset
print(data.describe())
```

HASIL

```
                age     anaemia  creatinine_phosphokinase    diabetes  \
count  299.000000  299.000000                299.000000  299.000000
mean    60.833893    0.431438                581.839465    0.418060
std     11.894809    0.496107                970.287881    0.494067
min     40.000000    0.000000                 23.000000    0.000000
25%     51.000000    0.000000                116.500000    0.000000
50%     60.000000    0.000000                250.000000    0.000000
75%     70.000000    1.000000                582.000000    1.000000
max     95.000000    1.000000               7861.000000    1.000000

       ejection_fraction  high_blood_pressure      platelets  \
count         299.000000           299.000000     299.000000
mean           38.083612             0.351171  263358.029264
std            11.834841             0.478136   97804.236869
min            14.000000             0.000000   25100.000000
25%            30.000000             0.000000  212500.000000
50%            38.000000             0.000000  262000.000000
75%            45.000000             1.000000  303500.000000
max            80.000000             1.000000  850000.000000

       serum_creatinine  serum_sodium         sex     smoking        time  \
count         299.00000    299.000000  299.000000  299.00000  299.000000
mean            1.39388    136.625418    0.648829    0.32107  130.260870
std             1.03451      4.412477    0.478136    0.46767   77.614208
min             0.50000    113.000000    0.000000    0.00000    4.000000
25%             0.90000    134.000000    0.000000    0.00000   73.000000
50%             1.10000    137.000000    1.000000    0.00000  115.000000
75%             1.40000    140.000000    1.000000    1.00000  203.000000
max             9.40000    148.000000    1.000000    1.00000  285.000000

       DEATH_EVENT
count    299.00000
mean       0.32107
std        0.46767
min        0.00000
25%        0.00000
50%        0.00000
75%        1.00000
max        1.00000
```

EXPLANATORY DATA ANALYSIS

CODE

```
# Memeriksa nilai yang hilang
print(data.isnull().sum())
```

HASIL

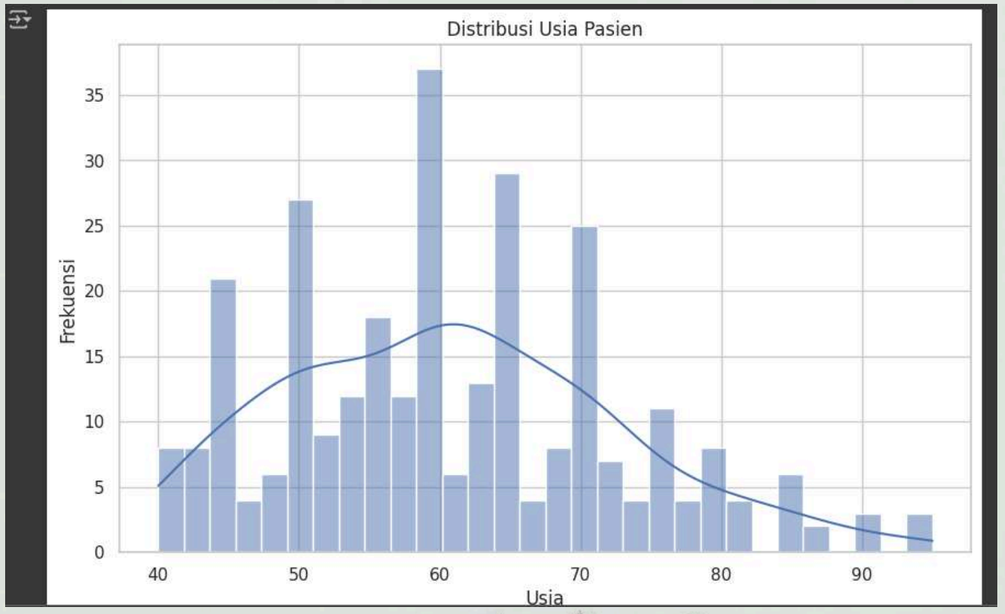
```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

DATA VISUALIZATION

CODE

```
[ ]  # Visualisasi distribusi usia pasien
     plt.figure(figsize=(10, 6))
     sns.histplot(data['age'], bins=30, kde=True)
     plt.title('Distribusi Usia Pasien')
     plt.xlabel('Usia')
     plt.ylabel('Frekuensi')
     plt.show()
```

HASIL

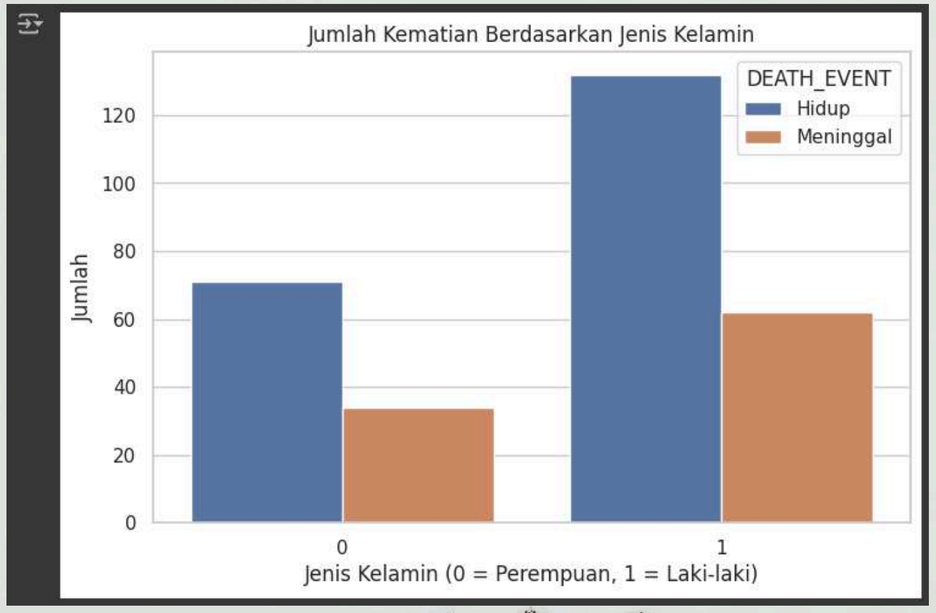


DATA VISUALIZATION

CODE

```
# Visualisasi jumlah kematian berdasarkan jenis kelamin
plt.figure(figsize=(8, 5))
sns.countplot(x='sex', hue='DEATH_EVENT', data=data)
plt.title('Jumlah Kematian Berdasarkan Jenis Kelamin')
plt.xlabel('Jenis Kelamin (0 = Perempuan, 1 = Laki-laki)')
plt.ylabel('Jumlah')
plt.legend(title='DEATH_EVENT', labels=['Hidup', 'Meninggal'])
plt.show()
```

HASIL

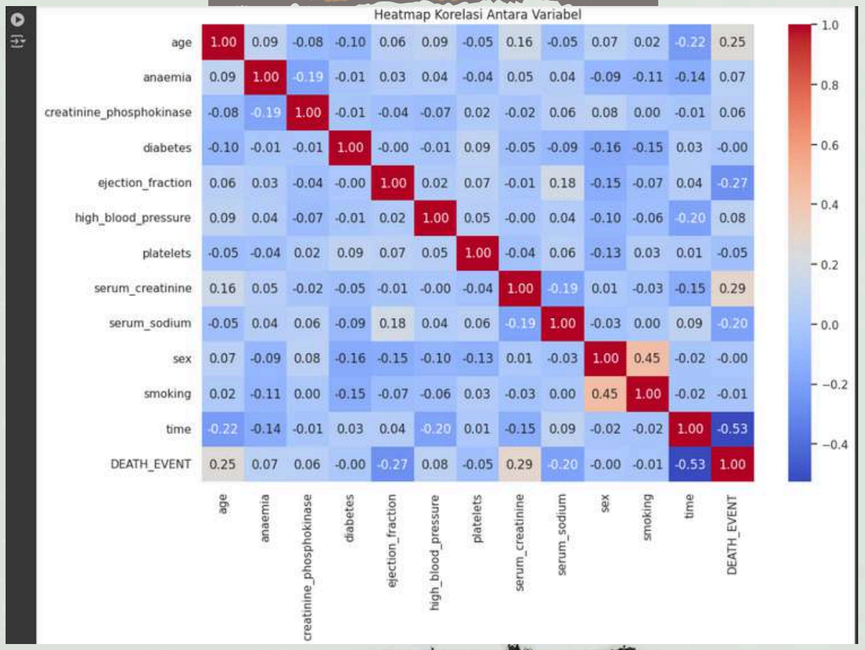


DATA VISUALIZATION

CODE

```python
# Visualisasi heatmap untuk melihat korelasi antar variabel
plt.figure(figsize=(12, 8))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Heatmap Korelasi Antara Variabel')
plt.show()
```

HASIL



DATA VISUALIZATION

CODE

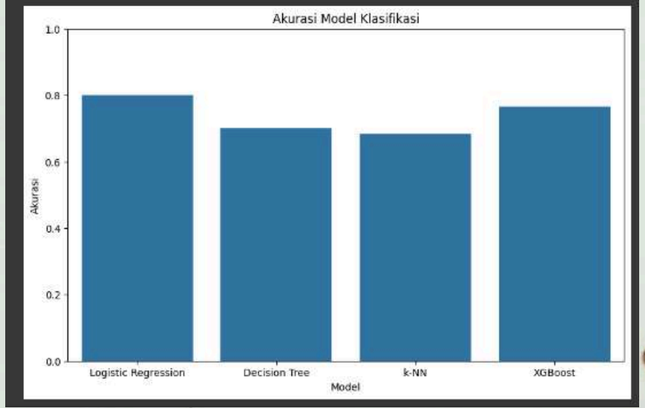
```python
# Import library yang dibutuhkan
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import warnings
warnings.filterwarnings('ignore')

# Memisahkan fitur dan target
X = data.drop('DEATH_EVENT', axis=1)  # Fitur
y = data['DEATH_EVENT']  # Target

# 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)

# Daftar model yang akan digunakan
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'k-NN': KNeighborsClassifier(),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss')
}

# Menyimpan hasil akurasi
results = {}
```

```python
# Loop untuk melatih dan menguji setiap model
for name, model in models.items():
    # Membuat pipeline
    pipeline = Pipeline([
        ('scaler', StandardScaler()),  # Normalisasi data
        ('classifier', model)  # Model klasifikasi
    ])

    # Melatih model
    pipeline.fit(X_train, y_train)

    # Memprediksi data uji
    y_pred = pipeline.predict(X_test)

    # Menghitung akurasi
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = accuracy

    # Menampilkan hasil klasifikasi
    print(f'--- {name} ---')
    print(f'Akurasi: {accuracy:.4f}')
    print('Confusion Matrix:')
    print(confusion_matrix(y_test, y_pred))
    print('Classification Report:')
    print(classification_report(y_test, y_pred))

# Visualisasi hasil akurasi
plt.figure(figsize=(10, 6))
sns.barplot(x=list(results.keys()), y=list(results.values()))
plt.title('Akurasi Model Klasifikasi')
plt.xlabel('Model')
plt.ylabel('Akurasi')
plt.ylim(0, 1)
plt.show()
```

HASIL

```
--- Logistic Regression ---
Akurasi: 0.8000
Confusion Matrix:
[[34  1]
 [11 14]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.97      0.85        35
           1       0.93      0.56      0.70        25

    accuracy                           0.80        60
   macro avg       0.84      0.77      0.77        60
weighted avg       0.83      0.80      0.79        60

--- Decision Tree ---
Akurasi: 0.7000
Confusion Matrix:
[[28  7]
 [11 14]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.80      0.76        35
           1       0.67      0.56      0.61        25

    accuracy                           0.70        60
   macro avg       0.69      0.68      0.68        60
weighted avg       0.70      0.70      0.70        60

--- k-NN ---
Akurasi: 0.6833
Confusion Matrix:
[[34  1]
 [18  7]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.97      0.78        35
           1       0.88      0.28      0.42        25

    accuracy                           0.68        60
   macro avg       0.76      0.63      0.60        60
weighted avg       0.75      0.68      0.63        60
```

```
--- XGBoost ---
Akurasi: 0.7667
Confusion Matrix:
[[31  4]
 [10 15]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.89      0.82        35
           1       0.79      0.60      0.68        25

    accuracy                           0.77        60
   macro avg       0.77      0.74      0.75        60
weighted avg       0.77      0.77      0.76        60
```



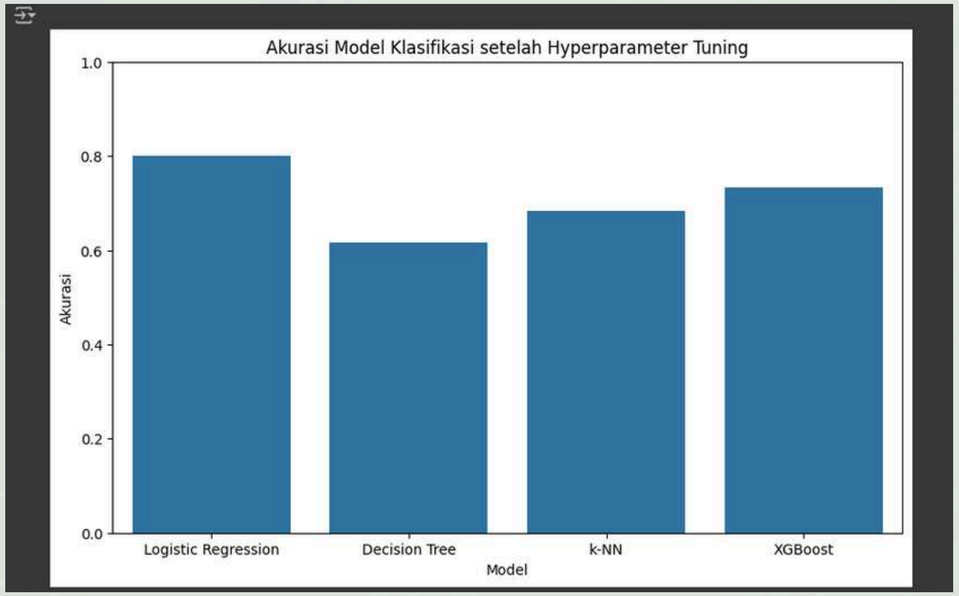
HYPERPARAMETER TUNING

CODE

```python
# Import library yang dibutuhkan
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report
import warnings
warnings.filterwarnings('ignore')

# Daftar model dan parameter yang akan di-tune
models_params = {
    'Logistic Regression': {
        'model': LogisticRegression(max_iter=1000),
        'params': {
            'classifier__C': [0.01, 0.1, 1, 10, 100],
            'classifier__solver': ['liblinear', 'lbfgs']
        }
    },
    'Decision Tree': {
        'model': DecisionTreeClassifier(),
        'params': {
            'classifier__max_depth': [None, 10, 20, 30],
            'classifier__min_samples_split': [2, 5, 10]
        }
    },
    'k-NN': {
        'model': KNeighborsClassifier(),
        'params': {
            'classifier__n_neighbors': [3, 5, 7, 9],
            'classifier__weights': ['uniform', 'distance']
        }
    },
    'XGBoost': {
        'model': XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
        'params': {
            'classifier__max_depth': [3, 5, 7],
            'classifier__learning_rate': [0.01, 0.1, 0.2],
            'classifier__n_estimators': [50, 100, 150]
        }
    }
}
```

```python
# Menyimpan hasil tuning
results = {}

# Loop untuk melatih dan menguji setiap model dengan hyperparameter tuning
for name, model_info in models_params.items():
    # Membuat pipeline
    pipeline = Pipeline([
        ('scaler', StandardScaler()),  # Normalisasi data
        ('classifier', model_info['model'])  # Model klasifikasi
    ])

    # GridSearchCV untuk hyperparameter tuning
    grid_search = GridSearchCV(pipeline, model_info['params'], cv=5, scoring='accuracy', n_jobs=-1)
    grid_search.fit(X_train, y_train)

    # Memprediksi data uji
    y_pred = grid_search.predict(X_test)

    # Menghitung akurasi
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = {
        'best_params': grid_search.best_params_,
        'accuracy': accuracy,
        'classification_report': classification_report(y_test, y_pred, output_dict=True)
    }

# Menampilkan hasil tuning
for name, result in results.items():
    print(f'--- {name} ---')
    print(f'Best Parameters: {result["best_params"]}')
    print(f'Akurasi: {result["accuracy"]:.4f}')
    print('Classification Report:')
    print(result['classification_report'])
    print('\n' + '-'*50 + '\n')

# Visualisasi hasil akurasi
model_names = list(results.keys())
accuracies = [result['accuracy'] for result in results.values()]

plt.figure(figsize=(10, 6))
sns.barplot(x=model_names, y=accuracies)
plt.title('Akurasi Model Klasifikasi setelah Hyperparameter Tuning')
plt.xlabel('Model')
plt.ylabel('Akurasi')
plt.ylim(0, 1)
plt.show()
```

HASIL



CLASSIFICATION MODE

EXPLANATORY DATA ANALYSIS

CODE

```python
# Mengimpor library yang diperlukan
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Membaca dataset dari file CSV
data = pd.read_csv('student-por.csv', sep=';')

# Menampilkan 5 baris pertama dari dataset
print("5 Baris Pertama Dataset:")
print(data.head())
```

HASIL

```
5 Baris Pertama Dataset:
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  ... \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  ...
1     GP   F   17       U     GT3       T     1     1  at_home     other  ...
2     GP   F   15       U     LE3       T     1     1  at_home     other  ...
3     GP   F   15       U     GT3       T     4     2   health  services  ...
4     GP   F   16       U     GT3       T     3     3    other     other  ...

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        4   0  11  11
1      5        3      3     1     1      3        2   9  11  11
2      4        3      2     2     3      3        6  12  13  12
3      3        2      2     1     1      5        0  14  14  14
4      4        3      2     1     2      5        0  11  13  13

[5 rows x 33 columns]
```

EXPLANATORY DATA ANALYSIS

CODE

```python
# Menampilkan informasi dasar tentang dataset
print("\nInformasi Dataset:")
print(data.info())
```

HASIL

```
Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64 
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64 
 7   Fedu        649 non-null    int64 
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64 
 13  studytime   649 non-null    int64 
 14  failures    649 non-null    int64 
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64 
 24  freetime    649 non-null    int64 
 25  goout       649 non-null    int64 
 26  Dalc        649 non-null    int64 
 27  Walc        649 non-null    int64 
 28  health      649 non-null    int64 
 29  absences    649 non-null    int64 
 30  G1          649 non-null    int64 
 31  G2          649 non-null    int64 
 32  G3          649 non-null    int64 
dtypes: int64(16), object(17)
memory usage: 167.4+ KB
None
```

EXPLANATORY DATA ANALYSIS

CODE

```python
# Menampilkan deskripsi statistik dari dataset
print("\nDeskripsi Statistik:")
print(data.describe())
```

HASIL

```
Deskripsi Statistik:
                age        Medu        Fedu  traveltime   studytime    failures  \
count  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000
mean    16.744222    2.514638    2.306626    1.568567    1.930663    0.221880
std      1.218138    1.134552    1.099931    0.748660    0.829510    0.593235
min     15.000000    0.000000    0.000000    1.000000    1.000000    0.000000
25%     16.000000    2.000000    1.000000    1.000000    1.000000    0.000000
50%     17.000000    2.000000    2.000000    1.000000    2.000000    0.000000
75%     18.000000    4.000000    3.000000    2.000000    2.000000    0.000000
max     22.000000    4.000000    4.000000    4.000000    4.000000    3.000000

            famrel    freetime       goout        Dalc        Walc      health  \
count  649.000000  649.000000  649.000000  649.000000  649.000000  649.000000
mean     3.930663    3.180277    3.184900    1.502311    2.280431    3.536210
std      0.955717    1.051093    1.175766    0.924834    1.284380    1.446259
min      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
25%      4.000000    3.000000    2.000000    1.000000    1.000000    2.000000
50%      4.000000    3.000000    3.000000    1.000000    2.000000    4.000000
75%      5.000000    4.000000    4.000000    2.000000    3.000000    5.000000
max      5.000000    5.000000    5.000000    5.000000    5.000000    5.000000

          absences          G1          G2          G3
count  649.000000  649.000000  649.000000  649.000000
mean     3.659476   11.399076   11.570108   11.906009
std      4.640759    2.745265    2.913639    3.230656
min      0.000000    0.000000    0.000000    0.000000
25%      0.000000   10.000000   10.000000   10.000000
50%      2.000000   11.000000   11.000000   12.000000
75%      6.000000   13.000000   13.000000   14.000000
max     32.000000   19.000000   19.000000   19.000000
```

EXPLANATORY DATA ANALYSIS

CODE

```
# Menampilkan jumlah nilai null di setiap kolom
print("\nJumlah Nilai Null di Setiap Kolom:")
print(data.isnull().sum())
```

HASIL

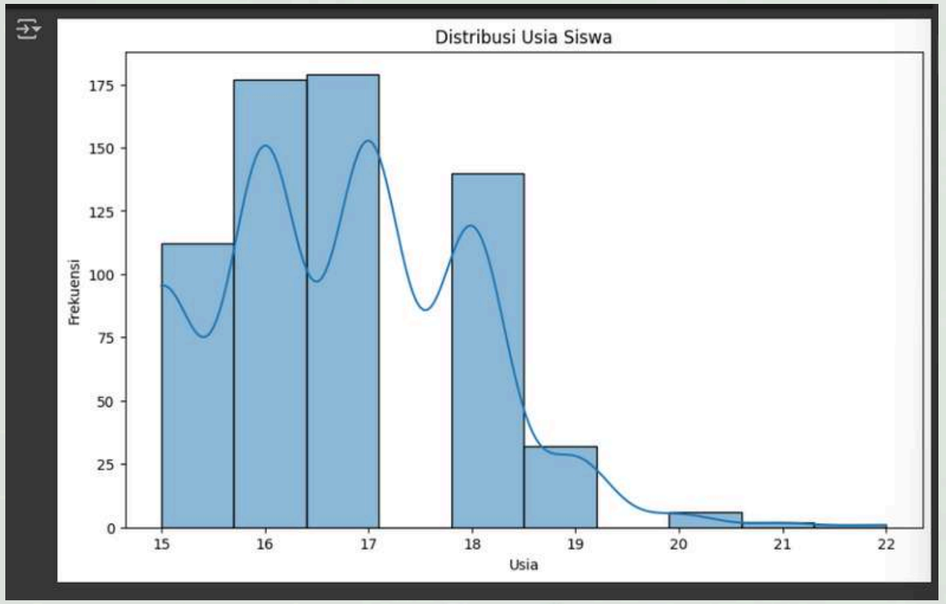
```
Jumlah Nilai Null di Setiap Kolom:
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
```

DATA VISUALIZATION

CODE

```
[9]  # Visualisasi distribusi usia
     plt.figure(figsize=(10, 6))
     sns.histplot(data['age'], bins=10, kde=True)
     plt.title('Distribusi Usia Siswa')
     plt.xlabel('Usia')
     plt.ylabel('Frekuensi')
     plt.show()
```

HASIL

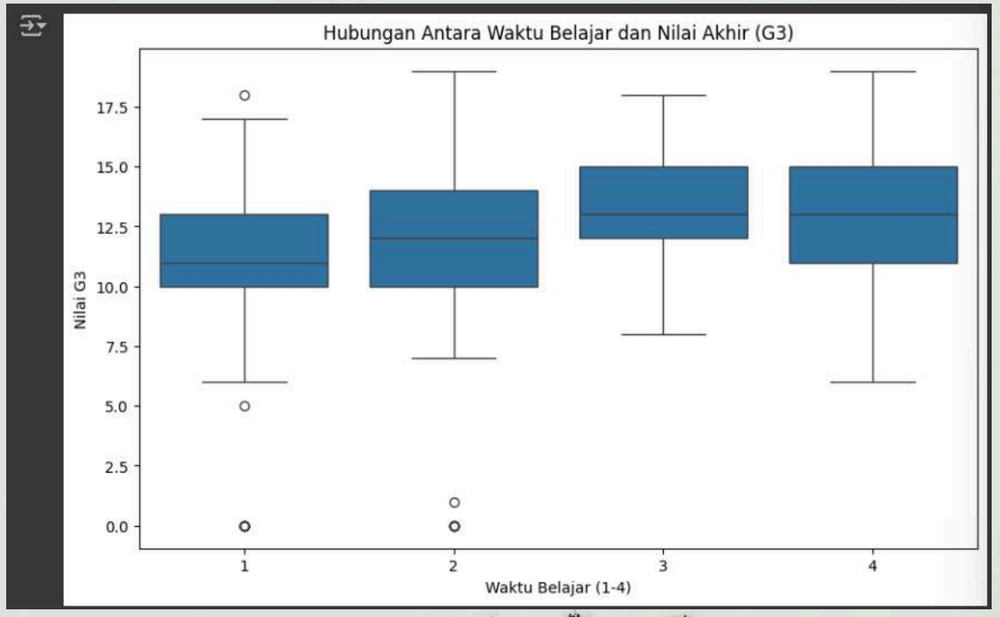


DATA VISUALIZATION

CODE

```
# Visualisasi hubungan antara waktu belajar dan nilai G3
plt.figure(figsize=(10, 6))
sns.boxplot(x='studytime', y='G3', data=data)
plt.title('Hubungan Antara Waktu Belajar dan Nilai Akhir (G3)')
plt.xlabel('Waktu Belajar (1-4)')
plt.ylabel('Nilai G3')
plt.show()
```

HASIL

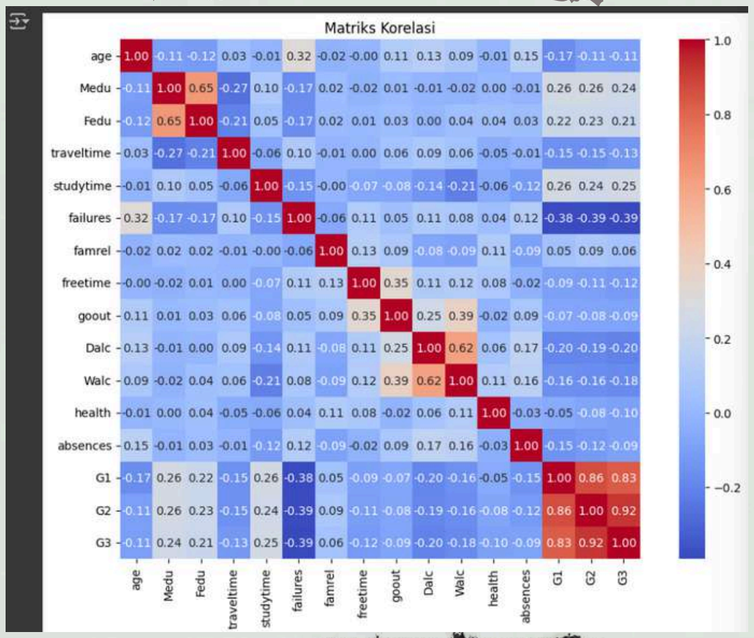


DATA VISUALIZATION

CODE

```
# Menampilkan korelasi antara fitur-fitur numerik
plt.figure(figsize=(12, 8))
# Mengambil hanya kolom numerik untuk matriks korelasi
correlation_matrix = data.select_dtypes(include=['float64', 'int64']).corr()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True)
plt.title('Matriks Korelasi')
plt.show()
```

HASIL



PIPELINE

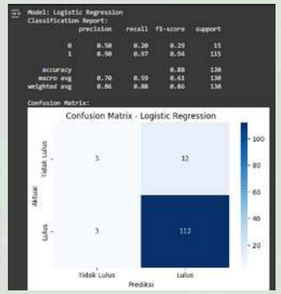
CODE

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Membaca dataset dari file CSV
data = pd.read_csv('student-por.csv', sep=';')

# Mengganti nilai kategorikal dengan angka (encoding)
data['sex'] = data['sex'].map({'F': 0, 'M': 1})
data['famsize'] = data['famsize'].map({'LE3': 0, 'GT3': 1})
data['Pstatus'] = data['Pstatus'].map({'T': 1, 'A': 0})

# Pilih fitur dan target
X = data[['age', 'sex', 'famsize', 'Pstatus', 'studytime', 'failures', 'absences']]
y = data['G3'].apply(lambda x: 1 if x >= 10 else 0)  # Klasifikasi biner: lulus (1) atau tidak lulus (0)

# Membagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Membuat pipeline untuk setiap model
models = {
    "Logistic Regression": Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', LogisticRegression(random_state=42))
    ]),
    "Decision Tree": Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', DecisionTreeClassifier(random_state=42))
    ]),
    "K-Nearest Neighbors": Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', KNeighborsClassifier())
    ]),
    "XGBoost": Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', XGBClassifier(random_state=42, eval_metric='mlogloss'))
    ])
}
```

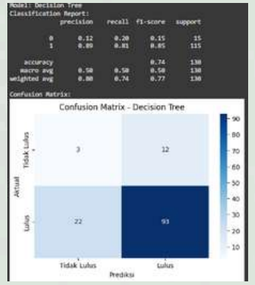
```python
# Menyimpan hasil akurasi untuk visualisasi
accuracy_scores = []

# Melatih dan mengevaluasi setiap model
for model_name, model_pipeline in models.items():
    # Melatih model
    model_pipeline.fit(X_train, y_train)

    # Prediksi pada data test
    y_pred = model_pipeline.predict(X_test)

    # Menghitung akurasi
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

    # Menampilkan classification report dan confusion matrix
    print(f"Model: {model_name}")
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print("Confusion Matrix:")
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Tidak Lulus', 'Lulus'], yticklabels=['Tidak Lulus', 'Lulus'])
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Prediksi')
    plt.ylabel('Aktual')
    plt.show()
```

HASIL

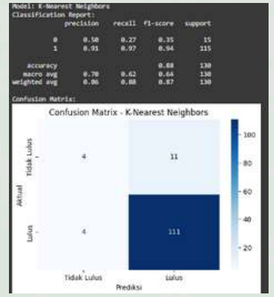
```
Model: Logistic Regression
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.20      0.29        15
           1       0.90      0.97      0.94       115

    accuracy                           0.88       130
   macro avg       0.70      0.59      0.61       130
weighted avg       0.86      0.88      0.86       130

Confusion Matrix:
```




```
Model: Decision Tree
Classification Report:
              precision    recall  f1-score   support

           0       0.12      0.20      0.15        15
           1       0.89      0.81      0.85       115

    accuracy                           0.74       130
   macro avg       0.50      0.50      0.50       130
weighted avg       0.80      0.74      0.77       130

Confusion Matrix:
```




```
Model: K-Nearest Neighbors
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.27      0.35        15
           1       0.91      0.97      0.94       115

    accuracy                           0.88       130
   macro avg       0.70      0.62      0.64       130
weighted avg       0.86      0.88      0.87       130

Confusion Matrix:
```




```
Model: XGBoost
Classification Report:
              precision    recall  f1-score   support

           0       0.25      0.27      0.26        15
           1       0.90      0.90      0.90       115

    accuracy                           0.82       130
   macro avg       0.58      0.58      0.58       130
weighted avg       0.83      0.82      0.83       130

Confusion Matrix:
```

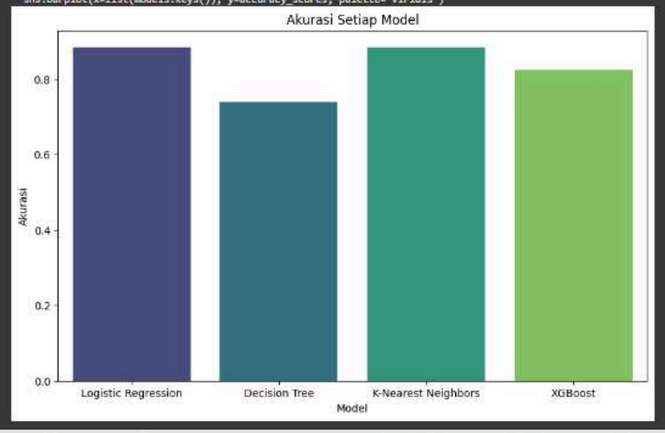
PIPELINE

CODE

```
# Visualisasi Akurasi Model
plt.figure(figsize=(10, 6))
sns.barplot(x=list(models.keys()), y=accuracy_scores, palette='viridis')
plt.title('Akurasi Setiap Model')
plt.xlabel('Model')
plt.ylabel('Akurasi')
plt.show()
```

HASIL

```
<ipython-input-13-d83626456b1d>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(models.keys()), y=accuracy_scores, palette='viridis')
```



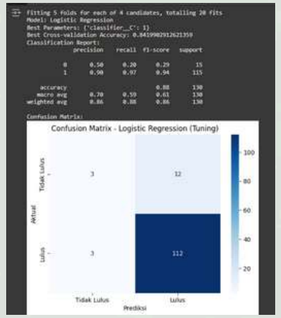
HYPERPARAMETER TUNING

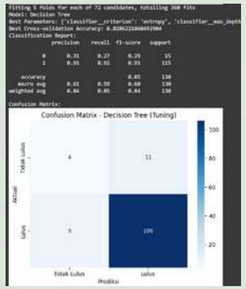
CODE

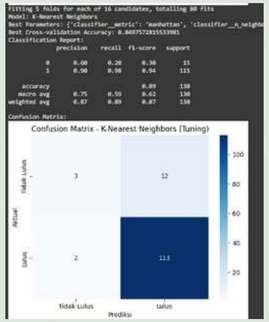
```python
from sklearn.model_selection import GridSearchCV

# Parameter grid untuk setiap model
param_grids = {
    "Logistic Regression": {
        'classifier__C': [0.1, 1, 10, 100]
    },
    "Decision Tree": {
        'classifier__max_depth': [3, 5, 7, 10],
        'classifier__min_samples_split': [2, 5, 10],
        'classifier__min_samples_leaf': [1, 2, 4],
        'classifier__criterion': ['gini', 'entropy']
    },
    "K-Nearest Neighbors": {
        'classifier__n_neighbors': [3, 5, 7, 10],
        'classifier__weights': ['uniform', 'distance'],
        'classifier__metric': ['euclidean', 'manhattan']
    },
    "XGBoost": {
        'classifier__learning_rate': [0.01, 0.1, 0.2],
        'classifier__n_estimators': [100, 200],
        'classifier__max_depth': [3, 5, 7],
        'classifier__colsample_bytree': [0.3, 0.5, 0.7]
    }
}
```

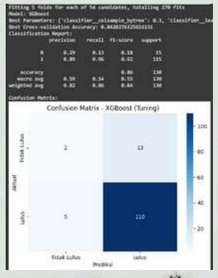
```python
# Menyimpan hasil akurasi untuk visualisasi
best_accuracy_scores = []
best_params = []

# Melakukan Grid Search untuk setiap model
for model_name, model_pipeline in models.items():
    # Membuat GridSearchCV
    grid_search = GridSearchCV(estimator=model_pipeline, param_grid=param_grids[model_name],
                               cv=5, n_jobs=-1, verbose=1)

    # Melatih model dengan GridSearchCV
    grid_search.fit(X_train, y_train)

    # Menyimpan hasil terbaik
    best_accuracy_scores.append(grid_search.best_score_)
    best_params.append(grid_search.best_params_)

    # Menampilkan hasil terbaik
    print(f"Model: {model_name}")
    print("Best Parameters:", grid_search.best_params_)
    print("Best Cross-validation Accuracy:", grid_search.best_score_)

    # Evaluasi pada data test menggunakan model terbaik
    y_pred = grid_search.best_estimator_.predict(X_test)

    # Menampilkan classification report dan confusion matrix
    print("Classification Report:")
    print(classification_report(y_test, y_pred))

    print("Confusion Matrix:")
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Tidak Lulus', 'Lulus'], yticklabels=['Tidak Lulus', 'Lulus'])
    plt.title(f'Confusion Matrix - {model_name} (Tuning)')
    plt.xlabel('Prediksi')
    plt.ylabel('Aktual')
```

HASIL









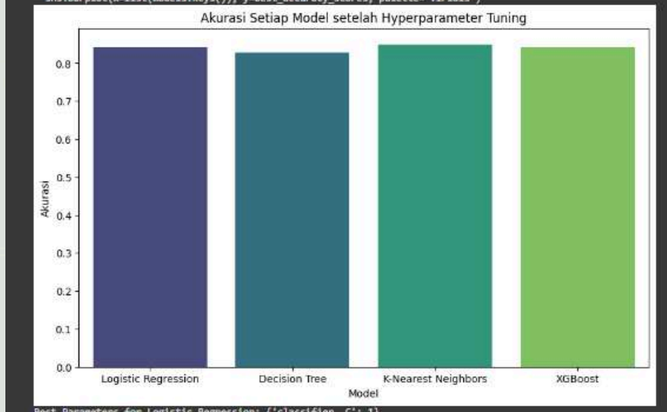
HYPERPARAMETER TUNING

CODE

```
# Visualisasi Hasil Akurasi setelah Tuning
plt.figure(figsize=(10, 6))
sns.barplot(x=list(models.keys()), y=best_accuracy_scores, palette='viridis')
plt.title('Akurasi Setiap Model setelah Hyperparameter Tuning')
plt.xlabel('Model')
plt.ylabel('Akurasi')
plt.show()

# Menampilkan parameter terbaik
for model_name, params in zip(models.keys(), best_params):
    print(f"Best Parameters for {model_name}: {params}")
```

HASIL

```
<ipython-input-15-1a2f5c548177>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=list(models.keys()), y=best_accuracy_scores, palette='viridis')
```




```
Best Parameters for Logistic Regression: {'classifier__C': 1}
Best Parameters for Decision Tree: {'classifier__criterion': 'entropy', 'classifier__max_depth': 3, 'classifier__min_samples_leaf': 4, 'classifier__min_samples_split': 2}
Best Parameters for K-Nearest Neighbors: {'classifier__metric': 'manhattan', 'classifier__n_neighbors': 10, 'classifier__weights': 'uniform'}
Best Parameters for XGBoost: {'classifier__colsample_bytree': 0.3, 'classifier__learning_rate': 0.1, 'classifier__max_depth': 7, 'classifier__n_estimators': 100}
```

REGRESSION MODEL

EXPLANATORY DATA ANALYSIS

CODE

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

HASIL

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

EXPLANATORY DATA ANALYSIS

CODE

```
!pip install pandas matplotlib seaborn plotly
```

HASIL

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.24.1)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

EXPLANATORY DATA ANALYSIS

CODE

```
[ ]  # Import pustaka
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import plotly.express as px
     from sklearn.pipeline import Pipeline
     from sklearn.preprocessing import PolynomialFeatures, StandardScaler
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.neighbors import KNeighborsRegressor
     from sklearn.linear_model import LinearRegression
     from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
     from xgboost import XGBRegressor
     from sklearn.metrics import mean_squared_error, r2_score
```

```
     # Memuat dataset dari Google Drive
     df = pd.read_csv('/content/drive/MyDrive/Dataset/RegresiUTSTelkom.csv')
```

```
[ ]  # Melihat 5 baris pertama data
     df.head()
```

HASIL

	2001	49.94357	21.47114	73.0775	8.74861	-17.40628	-13.09985	-25.01202	-12.23257	7.83089	...	13.0162	-54.40548	58.99367	15.37344	1.11144	-23.08793	68.40795	-1.82223	-27.46348	2.26327
0	2001	48.73215	18.42930	70.32679	12.94636	-10.32437	-24.83777	8.76630	-0.92019	18.76548	...	5.66812	-19.68073	33.04964	42.87836	-9.90378	-32.22788	70.49388	12.04941	58.43453	26.92061
1	2001	50.95714	31.85602	55.81851	13.41693	-6.57898	-18.54940	-3.27872	-2.35035	16.07017	...	3.03800	26.05866	-50.92779	10.93792	-0.07568	43.20130	-115.00698	-0.05859	39.67068	-0.66345
2	2001	48.24750	-1.89837	36.29772	2.58776	0.97170	-26.21683	5.05097	-10.34124	3.55005	...	34.57337	-171.70734	-16.96705	-46.67617	-12.51516	82.58061	-72.08993	9.90558	199.62971	10.85382
3	2001	50.97020	42.20998	67.09964	8.46791	-15.85279	-16.81409	-12.48207	-9.37636	12.63699	...	9.92661	-55.95724	64.92712	-17.72522	-1.49237	-7.50035	51.76631	7.88713	55.66926	28.74903
4	2001	50.54767	0.31568	92.35066	22.38696	-25.51870	-19.04928	20.67345	-5.19943	3.63566	...	6.59753	-50.69577	26.02574	18.94430	-0.33730	6.09352	35.18381	5.00283	-11.02257	0.02263

5 rows × 91 columns

EXPLANATORY DATA ANALYSIS

CODE

```
# Melihat informasi data seperti tipe data dan nilai kosong
df.info()
```

HASIL

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 515344 entries, 0 to 515343
Data columns (total 91 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   2001        515344 non-null  int64
 1   49.94357    515344 non-null  float64
 2   21.47114    515344 non-null  float64
 3   73.0775     515344 non-null  float64
 4   8.74861     515344 non-null  float64
 5   -17.40628   515344 non-null  float64
 6   -13.09905   515344 non-null  float64
 7   -25.01202   515344 non-null  float64
 8   -12.23257   515344 non-null  float64
 9   7.83089     515344 non-null  float64
 10  -2.46783    515344 non-null  float64
 11  3.32136     515344 non-null  float64
 12  -2.31521    515344 non-null  float64
 13  10.20556    515344 non-null  float64
 14  611.10913   515344 non-null  float64
 15  951.0896    515344 non-null  float64
 16  698.11428   515344 non-null  float64
 17  408.98485   515344 non-null  float64
 18  383.70912   515344 non-null  float64
 19  326.51512   515344 non-null  float64
 20  238.11327   515344 non-null  float64
 21  251.42414   515344 non-null  float64
 22  187.17351   515344 non-null  float64
```

EXPLANATORY DATA ANALYSIS

CODE

```
# Statistik deskriptif
df.describe()
```

HASIL

	73.0775	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	7.83089	...	13.0162	-54.40548	58.99367	15.37344	1.11544	-23.08793	68.40795	-1.82223	-27.46348	2.26327
515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	...	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000	515344.000000
8.658222	1.164110	-6.563580	-9.521968	-2.391046	-1.793215	3.727868	...	15.755411	-73.461537	41.542388	37.934163	0.315750	17.669292	-26.315520	4.456653	28.035229	1.329104
35.268505	16.322802	22.860803	12.857763	14.571853	7.963822	10.582869	...	32.099666	175.619058	122.228915	95.050718	16.161788	114.428002	173.977455	13.346567	185.558415	22.088598
-301.005060	-154.183580	-181.953370	-81.794290	-188.214000	-72.583850	-126.479040	...	-437.722830	-4402.376440	-1810.689190	-3098.350310	-341.789120	-3168.924570	-4319.992320	-236.039260	-7458.378150	-381.424430
-11.462775	-8.487587	-20.666455	-18.441005	-10.780360	-6.468390	-2.293670	...	-1.812650	-139.555737	-20.987115	-4.669655	-6.781598	-31.580617	-101.530305	-2.566137	-59.589453	-8.828248
10.476235	-0.652855	-6.007778	-11.188355	-2.046525	-1.736415	3.822305	...	9.171850	-53.089115	28.790580	33.623015	0.820830	15.598520	-21.204225	3.117645	7.750910	0.053015
29.764685	8.787548	7.741877	-2.388945	6.580587	2.913455	9.961865	...	26.274487	13.478793	89.661785	77.785810	8.471000	67.795510	52.389322	9.967742	86.351645	9.679640
322.851430	335.771820	262.868870	166.236890	172.402680	126.741270	146.297950	...	840.973380	4469.454870	3210.701700	1734.079690	260.544900	3662.865650	2833.608960	463.419500	7393.390440	677.899630

EXPLANATORY DATA ANALYSIS

CODE

```python
# Mengecek missing values
missing_values = df.isnull().sum()
print(f"Missing values in each column: \n{missing_values}")
```

HASIL

```
Missing values in each column:
2001          0
49.94357      0
21.47114      0
73.0775       0
8.74861       0
             ..
-23.08793     0
68.40795      0
-1.82223      0
-27.46348     0
2.26327       0
Length: 91, dtype: int64
```

EXPLANATORY DATA ANALYSIS

CODE

```
# Tampilkan daftar kolom dalam dataset
df.columns
```

HASIL

```
Index(['2001', '49.94357', '21.47114', '73.0775', '8.74861', '-17.40628',
       '-13.09905', '-25.01202', '-12.23257', '7.83089', '-2.46783', '3.32136',
       '-2.31521', '10.20556', '611.10913', '951.0896', '698.11428',
       '408.98485', '383.70912', '326.51512', '238.11327', '251.42414',
       '187.17351', '100.42652', '179.19498', '-8.41558', '-317.87038',
       '95.86266', '48.10259', '-95.66303', '-18.06215', '1.96984', '34.42438',
       '11.7267', '1.3679', '7.79444', '-0.36994', '-133.67852', '-83.26165',
       '-37.29765', '73.04667', '-37.36684', '-3.13853', '-24.21531',
       '-13.23066', '15.93809', '-18.60478', '82.15479', '240.5798',
       '-10.29407', '31.58431', '-25.38187', '-3.90772', '13.29258', '41.5506',
       '-7.26272', '-21.00863', '105.50848', '64.29856', '26.08481',
       '-44.5911', '-8.30657', '7.93706', '-10.7366', '-95.44766', '-82.03307',
       '-35.59194', '4.69525', '70.95626', '28.09139', '6.02015', '-37.13767',
       '-41.1245', '-8.40816', '7.19877', '-8.60176', '-5.90857', '-12.32437',
       '14.68734', '-54.32125', '40.14786', '13.0162', '-54.40548', '58.99367',
       '15.37344', '1.11144', '-23.08793', '68.40795', '-1.82223', '-27.46348',
       '2.26327'],
      dtype='object')
```

EXPLANATORY DATA ANALYSIS

CODE

```
# Menentukan kolom target yang sesuai (misalnya kolom yang paling akhir)
target_column = df.columns[-1]
```

```
# Tampilkan informasi statistik untuk kolom target
print(f"Informasi Kolom Target ({target_column}):")
print(df[target_column].describe())
```

HASIL

```
Informasi Kolom Target (2.26327):
count    515344.000000
mean          1.329104
std          22.088598
min        -381.424430
25%          -8.820248
50%           0.053015
75%           9.679540
max         677.899630
Name: 2.26327, dtype: float64
```

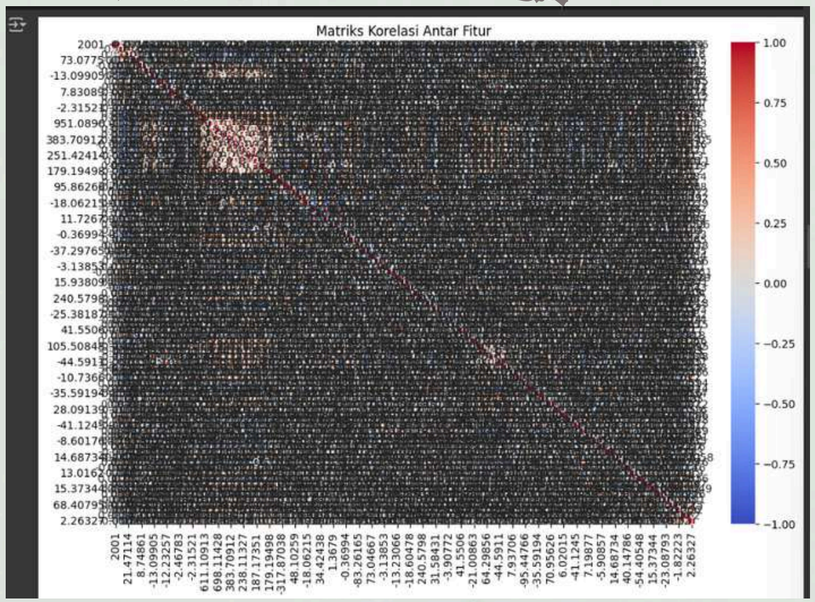
DATA VISUALIZATION

DATA VISUALIZATION

CODE

```python
# Menghitung korelasi antar fitur
correlation_matrix = df.corr()

# Visualisasi matriks korelasi menggunakan heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Matriks Korelasi Antar Fitur")
plt.show()
```

HASIL

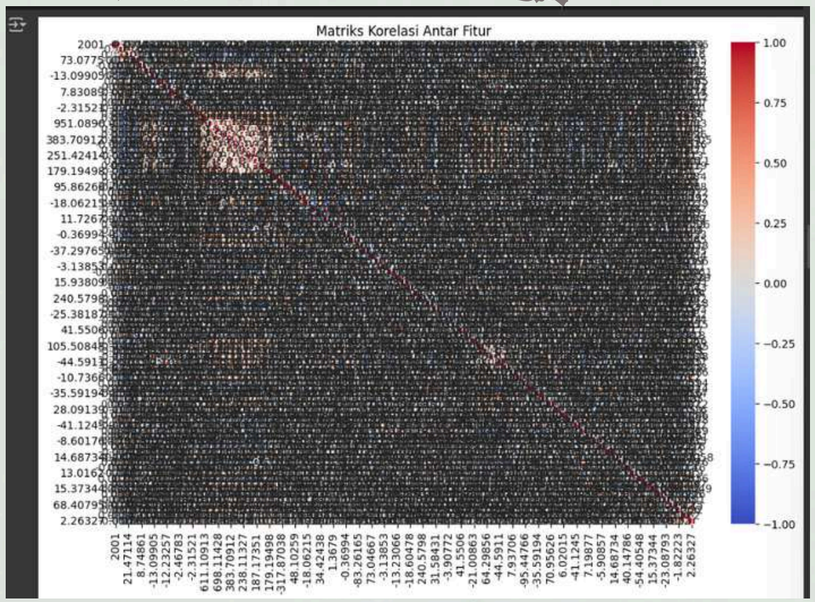


DATA VISUALIZATION

CODE

```python
# Menghitung korelasi antar fitur
correlation_matrix = df.corr()

# Visualisasi matriks korelasi menggunakan heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title("Matriks Korelasi Antar Fitur")
plt.show()
```

HASIL

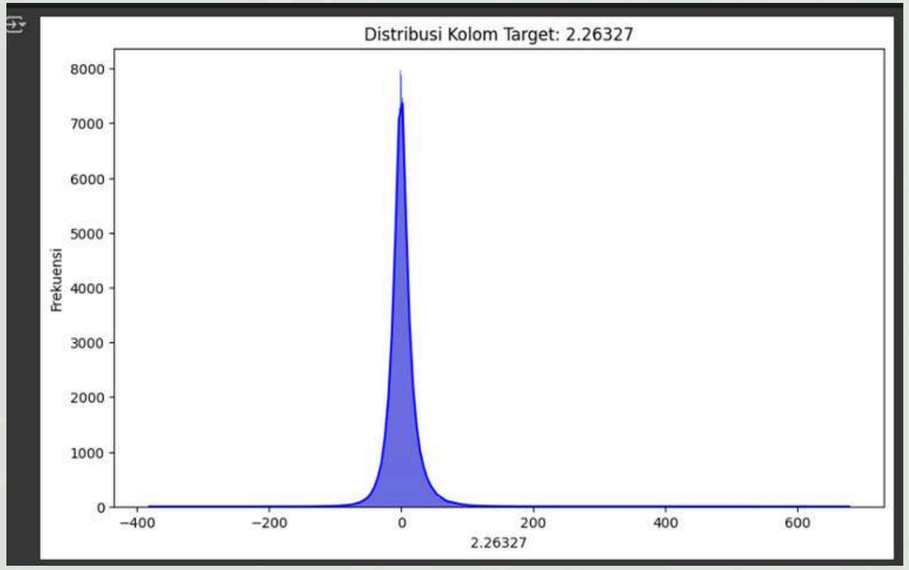


DATA VISUALIZATION

CODE

```python
# Visualisasi distribusi kolom target dengan histogram
plt.figure(figsize=(10, 6))
sns.histplot(df[target_column], kde=True, color='blue')
plt.title(f"Distribusi Kolom Target: {target_column}")
plt.xlabel(target_column)
plt.ylabel('Frekuensi')
plt.show()
```

HASIL

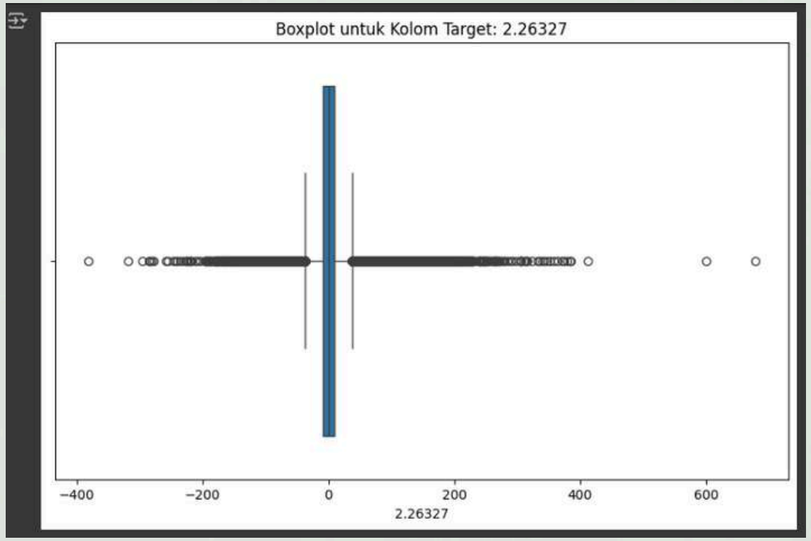


DATA VISUALIZATION

CODE

```
# Boxplot untuk mendeteksi outliers pada kolom target
plt.figure(figsize=(10, 6))
sns.boxplot(x=df[target_column])
plt.title(f"Boxplot untuk Kolom Target: {target_column}")
plt.show()
```

HASIL

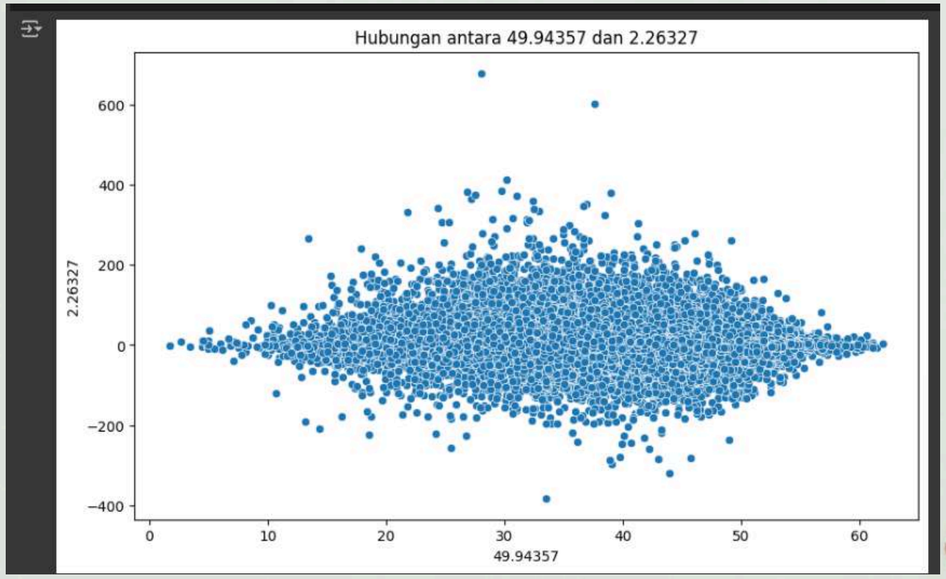


DATA VISUALIZATION

CODE

```
# Visualisasi hubungan antara fitur pertama dan target
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df[df.columns[1]], y=df[target_column])
plt.title(f"Hubungan antara {df.columns[1]} dan {target_column}")
plt.show()
```

HASIL



PIPELINE

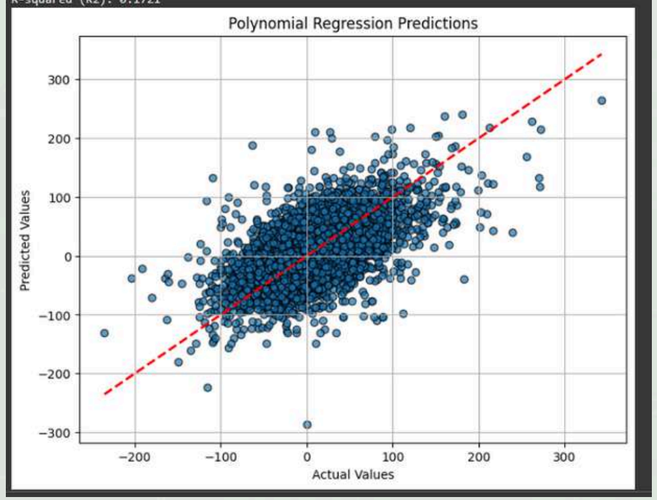
CODE

```
import matplotlib.pyplot as plt

# Fungsi untuk visualisasi
def visualize_predictions(model, X_test, y_test, title):
    predictions = model.predict(X_test)
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, predictions, alpha=0.7, edgecolors='k')
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
    plt.title(title)
    plt.xlabel("Actual Values")
    plt.ylabel("Predicted Values")
    plt.grid(True)
    plt.show()

# 1. Polynomial Regression
print("\n===== Polynomial Regression =====")
polynomial_pipeline = Pipeline([
    ('poly_features', PolynomialFeatures(degree=1)),
    ('scaler', StandardScaler()),
    ('regressor', DecisionTreeRegressor(random_state=42))
])
polynomial_pipeline.fit(X_train, y_train)
evaluate_model(polynomial_pipeline, X_test, y_test)
visualize_predictions(polynomial_pipeline, X_test, y_test, "Polynomial Regression Predictions")

# 2. Decision Tree Regression
print("\n===== Decision Tree Regression =====")
decision_tree_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', DecisionTreeRegressor(random_state=42))
])
decision_tree_pipeline.fit(X_train, y_train)
evaluate_model(decision_tree_pipeline, X_test, y_test)
visualize_predictions(decision_tree_pipeline, X_test, y_test, "Decision Tree Regression Predictions")
```

```
# 3. k-NN Regression
print("\n===== k-NN Regression =====")
knn_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', KNeighborsRegressor(n_neighbors=5))
])
knn_pipeline.fit(X_train, y_train)
evaluate_model(knn_pipeline, X_test, y_test)
visualize_predictions(knn_pipeline, X_test, y_test, "k-NN Regression Predictions")

# 4. XGBoost Regression
print("\n===== XGBoost Regression =====")
xgboost_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', XGBRegressor(
        objective='reg:squarederror',
        random_state=42,
        max_depth=3,
        n_estimators=50
    ))
])
xgboost_pipeline.fit(X_train, y_train)
evaluate_model(xgboost_pipeline, X_test, y_test)
visualize_predictions(xgboost_pipeline, X_test, y_test, "XGBoost Regression Predictions")

# Bersihkan memori
gc.collect()
```

HASIL

```
===== Polynomial Regression =====
Mean Squared Error (MSE): 403.0147
R-squared (R2): 0.1721
```



TERIMA KASIH