UTS ML

ANDREAS HASIHOLAN SINAGA

1103213033

REGRESSION MODEL

ABOUT DATASET

Dataset asli berisi sekitar 583.000 data, namun ukuran sebesar itu menyulitkan proses analisis di komputer dengan kapasitas pemrosesan terbatas. Untuk mengatasi hal ini, pengambilan sampel dilakukan secara acak sehingga menghasilkan dataset yang lebih kecil (15.000 baris), namun tetap mewakili data yang ada. Reduksi ini bertujuan untuk memastikan analisis dapat berjalan dengan efisien tanpa mengorbankan kualitas hasilnya.

STRUKTUR DATASET

Kolom 'Year': Menunjukkan tahun pelaksanaan UTS. Nilai dalam kolom ini mencakup berbagai tahun, seperti 1990-an hingga tahun-tahun terbaru.

Kolom 'x1' hingga 'x90': Kolom ini berisi nilai-nilai numerik yang mencerminkan variabel-variabel yang mungkin memengaruhi hasil UTS. Variabel ini dapat mencakup berbagai aspek akademik, kegiatan, atau faktor lain yang relevan. Nilai dalam kolom-kolom ini mencakup angka positif dan negatif dengan rentang yang cukup lebar.

MODELING

- Polynomial/Basis Function
- Decision Tree
- k-NN
- XGBoost Regression

POLYNOMIAL/BASIS FUNCTION

4. Hyperparameter Tuning

```python
# 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
}
```

[26] Python

```python
# Melakukan pencarian grid yang lebih terbatas
grid_search_limited = GridSearchCV(pipeline, param_grid_limited, cv=3, scoring='r2', verbose=2)
grid_search_limited.fit(X_train, y_train)
```

[27] Python

```
Fitting 3 folds for each of 2 candidates, totalling 6 fits
[CV] END linear_regression__fit_intercept=True, poly_features__degree=2; total time=  17.3s
[CV] END linear_regression__fit_intercept=True, poly_features__degree=2; total time=  16.0s
[CV] END linear_regression__fit_intercept=True, poly_features__degree=2; total time=  16.4s
[CV] END linear_regression__fit_intercept=True, poly_features__degree=3; total time=37.5min
[CV] END linear_regression__fit_intercept=True, poly_features__degree=3; total time=50.6min
[CV] END linear_regression__fit_intercept=True, poly_features__degree=3; total time=30.8min
```

GridSearchCV

estimator: Pipeline

StandardScaler

PolynomialFeatures

LinearRegression

REPORT
POLYNOMIAL/BASIS FUNCTION

```
Hyperparameter Terbaik (Versi Terbatas): {'linear_regression__fit_intercept': True, 'poly_features__degree': 2}
Mean Squared Error (MSE): 941.5384040822794
R-squared (R2): -7.324716040905317
```

Dari hasil modeling menggunakan modeling Polynomial dan menggunakan hyper parameter di dapat kan hasil hyperparameter tebaik yaitu {'linear_regression__fit_intercept': True, 'poly_features__degree': 2} dan mendapatkan data dia antara lain MSE(Mean Squared Error) sebesar 941.5384040822794 danR squared (R2): -7.324716040905317

DECISION TREE

4. Hyperparameter Tuning

```python
# Menentukan parameter untuk hyperparameter tuning
param_grid = {
    'decision_tree__max_depth': [None, 10, 20, 30],  # Kedalaman maksimal pohon
    'decision_tree__min_samples_split': [2, 5, 10],  # Minimum sampel untuk membagi node
    'decision_tree__min_samples_leaf': [1, 2, 4],  # Minimum sampel per daun
    'decision_tree__max_features': [None, 'sqrt', 'log2']  # Jumlah fitur untuk dipertimbangkan saat membelah
}
```

[36] Python

```python
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='r2', verbose=2)
grid_search.fit(X_train, y_train)
```

[37] Python

```
[CV] END decision_tree__max_depth=10, decision_tree__max_features=sqrt, decision_tree__min_samples_leaf=4, decision_tree__min_samples_split=10; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=sqrt, decision_tree__min_samples_leaf=4, decision_tree__min_samples_split=10; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=2; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=2; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=2; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=5; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=5; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=5; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=10; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=10; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=1, decision_tree__min_samples_split=10; total time=   0.0s
[CV] END decision_tree__max_depth=10, decision_tree__max_features=log2, decision_tree__min_samples_leaf=2, decision_tree__min_samples_split=2; total time=   0.0s
```

REPORT
DECISION TREE

```
Hyperparameter Terbaik: {'decision_tree__max_depth': 10, 'decision_tree__max_features': 'log2', 'decision_tree__min_samples_leaf': 4, 'decision_tree__min_samples_split': 10}
Mean Squared Error (MSE): 126.51729180093017
R-squared (R2): -0.11861664265694993
```

Dari hasil modeling menggunakan modeling Polynomial dan menggunakan hyper parameter di dapat kan hasil hyperparameter tebaik yaitu {'decision_tree__max_depth': 10, 'decision_tree__max_features': 'log2', 'decision_tree__min_samples_leaf': 4, 'decision_tree__min_samples_split': 10} dan mendapatkan data dia antara lain MSE(Mean Squared Error) sebesar 126.51729180093017dan R squared (R2): -0.11861664265694993

K-NN

4. 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
}
```

[46] Python

```
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='r2', verbose=2)
grid_search.fit(X_train, y_train)
```

[47] Python

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   2.6s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.2s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END .knn__n_neighbors=5, knn__p=1, knn__weights=uniform; total time=   0.2s
```

REPORT
K-NN

```
...  Hyperparameter Terbaik: {'knn__n_neighbors': 10, 'knn__p': 2, 'knn__weights': 'distance'}
     Mean Squared Error (MSE): 95.57613414300606
     R-squared (R2): 0.15495302838600844
```

Dari hasil modeling menggunakan modeling Polynomial dan menggunakan hyper parameter di dapat kan hasil hyperparameter tebaik yaitu {'knn__n_neighbors': 10, 'knn__p': 2, 'knn__weights': 'distance'} dan mendapatkan data dia antara lain MSE(Mean Squared Error) sebesar 95.57613414300606dan R squared (R2): 0.15495302838600844

XGBOOST REGRESSION

4. 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
}
```

[46] Python

```
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='r2', verbose=2)
grid_search.fit(X_train, y_train)
```

[47] Python

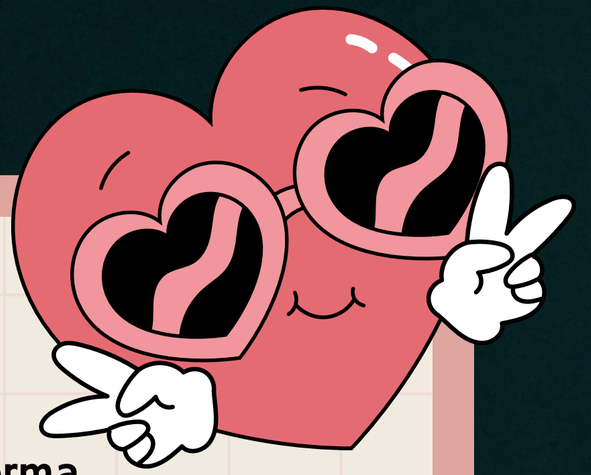
```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=  2.6s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=  0.2s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=  0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=  0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=  0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=  0.2s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=  0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=  0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=  0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=  0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=  0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=  0.0s
[CV] END .knn__n_neighbors=5, knn__p=1, knn__weights=uniform; total time=  0.2s
```

REPORT
XGBOOST REGRESSION

```
Hyperparameter Terbaik: {'xgb__colsample_bytree': 1.0, 'xgb__gamma': 0, 'xgb__learning_rate': 0.1, 'xgb__max_depth': 3, 'xgb__n_estimators': 300, 'xgb__reg_alpha': 0.1, 'xgb__reg_lambda': 10, 'xgb__subsample': 0.8}
Mean Squared Error (MSE): 81.05153672574461
R-squared (R2): 0.2833738645228231
```

Dari hasil modeling menggunakan modeling Polynomial dan menggunakan hyper parameter di dapat kan hasil hyperparameter tebaik yaitu {'knn__n_neighbors': 10, 'knn__p': 2, 'knn__weights': 'distance'} dan mendapatkan data dia antara lain MSE(Mean Squared Error) sebesar 81.05153672574461 dan R squared (R2): 0.2833738645228231

KESIMPULAN REGRESSION MODEL



Berdasarkan hasil evaluasi empat model, XGBoost memiliki performa terbaik dengan Mean Squared Error (MSE) terendah sebesar 81.05 dan nilai R-squared (R2) sebesar 0.28, menunjukkan kemampuannya menangkap pola data dengan baik. Model KNN berada di posisi kedua dengan MSE sebesar 95.58 dan R2 sebesar 0.15, diikuti oleh Decision Tree dengan MSE 126.52 dan R2 -0.12, yang menunjukkan prediksi kurang akurat. Linear Regression dengan Polynomial Degree 2 memiliki performa terburuk dengan MSE tertinggi (941.54) dan R2 negatif yang sangat rendah (-7.32), menandakan model ini tidak sesuai dengan data.

CLASSIFICATION MODEL

ABOUT DATASET

Dataset ini berisi tentang Evaluasi Mobil berasal dari model keputusan hierarki sederhana yang awalnya dikembangkan untuk demonstrasi DEX, M. Bohanec, V. Rajkovic: Sistem pakar untuk pengambilan keputusan. Sistemica 1(1), hal.145-157, 1990.)

STRUKTUR DATASET

Kolom	Deskripsi
buying	Harga beli
maint	Harga pemeliharaan
Year	Tahun pembuatan mobil
doors	Jumlah pintu
persons	Kapasitas jumlah orang yang dapat dibawa
lug_boot	Ukuran bagasi mobil
safety	Perkiraan tingkat keselamatan mobil
class	Tingkat evaluasi (tidak diterima, diterima, baik, sangat baik)

MODELING

- Logistics regession
- Decision Tree
- k-NN
- XGBoost Classfication

LOGISTICS REGESSION

4. Hyperparameter Tuning

```python
# 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
}
```

[46] Python

```python
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='r2', verbose=2)
grid_search.fit(X_train, y_train)
```

[47] Python

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   2.6s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.2s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END .knn__n_neighbors=5, knn__p=1, knn__weights=uniform; total time=   0.2s
```

REPORT
LOGISTICS REGESSION

Hyperparameter Terbaik: {'logistic_regression__C': 10, 'logistic_regression__penalty': 'l1', 'logistic_regression__solver': 'saga'}
Accuracy: 0.9364161849710982

Classification Report:

	precision	recall	f1-score	support
acc	0.91	0.83	0.87	83
good	0.67	0.91	0.77	11
unacc	0.96	0.97	0.97	235
vgood	0.94	0.94	0.94	17
accuracy			0.94	346
macro avg	0.87	0.91	0.89	346
weighted avg	0.94	0.94	0.94	346

DECISION TREE

4. 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
}
```

[46] Python

```
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='r2', verbose=2)
grid_search.fit(X_train, y_train)
```

[47] Python

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   2.6s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.2s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.2s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END .knn__n_neighbors=5, knn__p=1, knn__weights=uniform; total time=   0.2s
```

REPORT
DECISION TREE

Hyperparameter Terbaik: {'decision_tree__criterion': 'entropy', 'decision_tree__max_depth': None, 'decision_tree__min_samples_leaf': 1, 'decision_tree__min_samples_split': 2}
Accuracy: 0.9624277456647399

Classification Report:

	precision	recall	f1-score	support
acc	0.99	0.89	0.94	83
good	0.67	0.91	0.77	11
unacc	0.98	1.00	0.99	235
vgood	0.82	0.82	0.82	17
accuracy			0.96	346
macro avg	0.87	0.91	0.88	346
weighted avg	0.97	0.96	0.96	346

K-NN

5. Hyperparameter Tuning

```python
# 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
}
```

Python

```python
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='accuracy', verbose=2)
grid_search.fit(X_train, y_train)
```

Python

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.1s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.1s
[CV] END .knn__n_neighbors=3, knn__p=1, knn__weights=uniform; total time=   0.1s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=1, knn__weights=distance; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END .knn__n_neighbors=3, knn__p=2, knn__weights=uniform; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
[CV] END knn__n_neighbors=3, knn__p=2, knn__weights=distance; total time=   0.0s
```

Pieces Settings ⊗ 1 ⚠ 85 0 Connect Live Share BLACKBOX Chat Add Logs Improve Code Share Code Link Search Error Spaces: 4 Cell 4 of 131 Go Live AI Code Chat Codeium:

REPORT
K-NN

```
...
Hyperparameter Terbaik: {'knn__n_neighbors': 10, 'knn__p': 1, 'knn__weights': 'distance'}
Accuracy: 0.8930635838150289
```

Classification Report:

	precision	recall	f1-score	support
acc	0.87	0.72	0.79	83
good	0.55	0.55	0.55	11
unacc	0.91	1.00	0.95	235
vgood	0.90	0.53	0.67	17
accuracy			0.89	346
macro avg	0.81	0.70	0.74	346
weighted avg	0.89	0.89	0.89	346

XGBOOST CLASSFICATION

6. Hyperparameter Tuning

```python
# Menentukan parameter untuk hyperparameter tuning
param_grid = {
    'xgb__n_estimators': [100, 200],  # Jumlah estimator
    'xgb__max_depth': [3, 5, 7],  # Kedalaman maksimum pohon
    'xgb__learning_rate': [0.01, 0.1, 0.2],  # Kecepatan pembelajaran
    'xgb__subsample': [0.8, 1.0],  # Proporsi sampel yang digunakan untuk setiap pohon
    'xgb__colsample_bytree': [0.8, 1.0]  # Proporsi kolom yang digunakan untuk setiap pohon
}
```

Python

```python
# Melakukan pencarian grid untuk hyperparameter terbaik
grid_search = GridSearchCV(pipeline, param_grid, cv=3, scoring='accuracy', verbose=2)
grid_search.fit(X_train, y_train)
```

Python

```
  warnings.warn(smsg, UserWarning)
[CV] END xgb__colsample_bytree=1.0, xgb__learning_rate=0.2, xgb__max_depth=7, xgb__n_estimators=100, xgb__subsample=1.0; total time=   0.1s
[CV] END xgb__colsample_bytree=1.0, xgb__learning_rate=0.2, xgb__max_depth=7, xgb__n_estimators=100, xgb__subsample=1.0; total time=   0.1s
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [17:43:27] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [17:43:28] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

Pieces Settings | 1 85 | 0 | Connect | Live Share | BLACKBOX Chat | Add Logs | Improve Code | Share Code Link | Search Error | Spaces: 4 | Cell 4 of 131 | Go Live | AI Code Chat | Codeium:

REPORT
XGBOOST CLASSFICATION

Hyperparameter Terbaik: {'xgb__colsample_bytree': 1.0, 'xgb__learning_rate': 0.2, 'xgb__max_depth': 5, 'xgb__n_estimators': 200, 'xgb__subsample': 1.0}
Accuracy: 0.9739884393063584

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.92	0.95	83
1	0.58	1.00	0.73	11
2	1.00	1.00	1.00	235
3	1.00	0.88	0.94	17
accuracy			0.97	346
macro avg	0.89	0.95	0.91	346
weighted avg	0.98	0.97	0.98	346

KESIMPULAN CLASSIFICATION MODEL

Berdasarkan hasil evaluasi, XGBoost menunjukkan performa terbaik dengan akurasi 97.39% dan weighted average f1-score sebesar 0.98, menjadikannya model yang paling andal dalam menangani semua kelas dengan presisi dan recall yang tinggi. Decision Tree berada di urutan kedua dengan akurasi 96.24% dan weighted average f1-score 0.96, memberikan prediksi yang baik meski memiliki kelemahan pada kelas tertentu seperti "vgood". Logistic Regression mencatat akurasi 93.64% dengan weighted average f1-score 0.94, memberikan hasil stabil tetapi kurang optimal dalam memprediksi kelas "good" dengan recall 0.77. Sementara itu, KNN memiliki akurasi 89.30% dengan weighted average f1-score 0.89, tetapi kelemahan pada recall kelas tertentu membuatnya kurang ideal dibandingkan model lainnya.

TERIMA KASIH