

# Classification Model

**Raihana Fawaz (1103210102)**

# Car Evaluation Dataset

Car Evaluation Dataset adalah kumpulan data yang dirancang untuk analisis sistem pengambilan keputusan berbasis pembelajaran mesin. Dataset ini terdiri dari 1.728 instance dengan 6 fitur kategorikal, termasuk harga pembelian, biaya perawatan, jumlah pintu, kapasitas penumpang, ukuran bagasi, dan estimasi keamanan. Semua atribut memiliki nilai diskrit seperti vhigh, high, med, dan low, dengan target klasifikasi dalam empat kategori: unacc, acc, good, dan vgood.

Dataset ini lengkap tanpa nilai yang hilang, menjadikannya ideal untuk pengujian model pembelajaran mesin, khususnya klasifikasi. Car Evaluation Dataset sering digunakan untuk menguji metode induksi dan analisis struktur data karena struktur konsepnya sudah diketahui. Dataset ini cocok untuk eksplorasi analisis data dan pengembangan algoritma pengambilan keputusan.

# Import Library

```
[ ] import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from ucimlrepo import fetch_ucirepo
```

- **pandas as pd:** Digunakan untuk manipulasi data, seperti membaca dataset, membersihkan data, dan analisis tabel.
- **matplotlib.pyplot as plt:** Digunakan untuk membuat visualisasi grafik dasar, seperti diagram batang, garis, dan histogram.
- **seaborn as sns:** Digunakan untuk membuat visualisasi statistik yang lebih kompleks, seperti peta panas, distribusi data, atau visualisasi korelasi.
- **from ucimlrepo import fetch\_ucirepo:** Kemungkinan digunakan untuk mengambil dataset langsung dari UCI Machine Learning Repository melalui fungsi `fetch_ucirepo`.

# Menampilkan Dataset

```
# URL dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data"

# Kolom dataset (sesuai dokumentasi UCI)
column_names = [
    "buying", "maint", "doors", "persons", "lug_boot", "safety", "class"
]

# Muat dataset
car_evaluation = pd.read_csv(url, header=None, names=column_names)

# Tampilkan dataset
print(car_evaluation.head())
```

```
buying  maint  doors  persons  lug_boot  safety  class
0  vhigh   vhigh     2       2    small    low  unacc
1  vhigh   vhigh     2       2    small   med  unacc
2  vhigh   vhigh     2       2    small   high  unacc
3  vhigh   vhigh     2       2      med    low  unacc
4  vhigh   vhigh     2       2      med    med  unacc
```

# Tipe Data



```
# Menampilkan tipe data setiap kolom
print("\nTipe Data Kolom:")
print(dataset.dtypes)
```



```
Tipe Data Kolom:
buying      object
maint       object
doors        object
persons      object
lug_boot    object
safety       object
class        object
dtype: object
```

**Tipe data kategorikal ini perlu dikonversi ke bentuk numerik. Hal ini bisa dilakukan dengan teknik seperti one-hot encoding atau label encoding.**

# Cek Duplikasi

```
# Cek jumlah total baris duplikat
duplicate_rows = dataset[dataset.duplicated()]

# Tampilkan baris duplikat (jika ada)
print(f"Jumlah baris duplikat: {len(duplicate_rows)}")
if not duplicate_rows.empty:
    print("Baris duplikat:")
    print(duplicate_rows)

# Opsional: Menghapus duplikat (jika diperlukan)
df_no_duplicates = dataset.drop_duplicates()

# Konfirmasi penghapusan duplikat
print(f"Dataset setelah penghapusan duplikat: {df_no_duplicates.shape}")


```

→ Jumlah baris duplikat: 0  
Dataset setelah penghapusan duplikat: (1728, 7)

**Setelah konfirmasi dan penghapusan (meskipun tidak diperlukan), jumlah baris dataset tetap sebanyak 1.728 baris dengan 7 kolom. Hal ini menunjukkan integritas data tetap terjaga.**

# Cek Missing Values

```
# Mengecek Missing Values
print("Cek Missing Values:")
print(dataset.isnull().sum()) # Menampilkan jumlah nilai yang hilang per kolom

# Opsional: Menampilkan semua baris dengan nilai yang hilang
missing_rows = dataset[dataset.isnull().any(axis=1)]
print(f"\nJumlah baris dengan missing values: {len(missing_rows)}")
if not missing_rows.empty:
    print("Baris dengan nilai yang hilang:")
    print(missing_rows)

# Menghapus Missing Values
df_cleaned_rows = dataset.dropna()
print(f"\nDataset setelah menghapus baris dengan missing values: {df_cleaned_rows.shape}")

# Menghapus kolom yang memiliki nilai hilang
df_cleaned_columns = dataset.dropna(axis=1)
print(f"Dataset setelah menghapus kolom dengan missing values: {df_cleaned_columns.shape}")

# Opsional: Mengisi Missing Values
# Contoh: Mengisi nilai hilang dengan nilai rata-rata untuk kolom numerik
df_filled = dataset.fillna(dataset.mean(numeric_only=True))
print("\nDataset setelah mengisi nilai hilang dengan rata-rata:")
print(df_filled)
```

**Dataset mempertahankan atribut kategorikalnya dengan nilai seperti vhigh, low, small, dll., tanpa ada perubahan atau penghilangan informasi penting.**

# Data Visualization

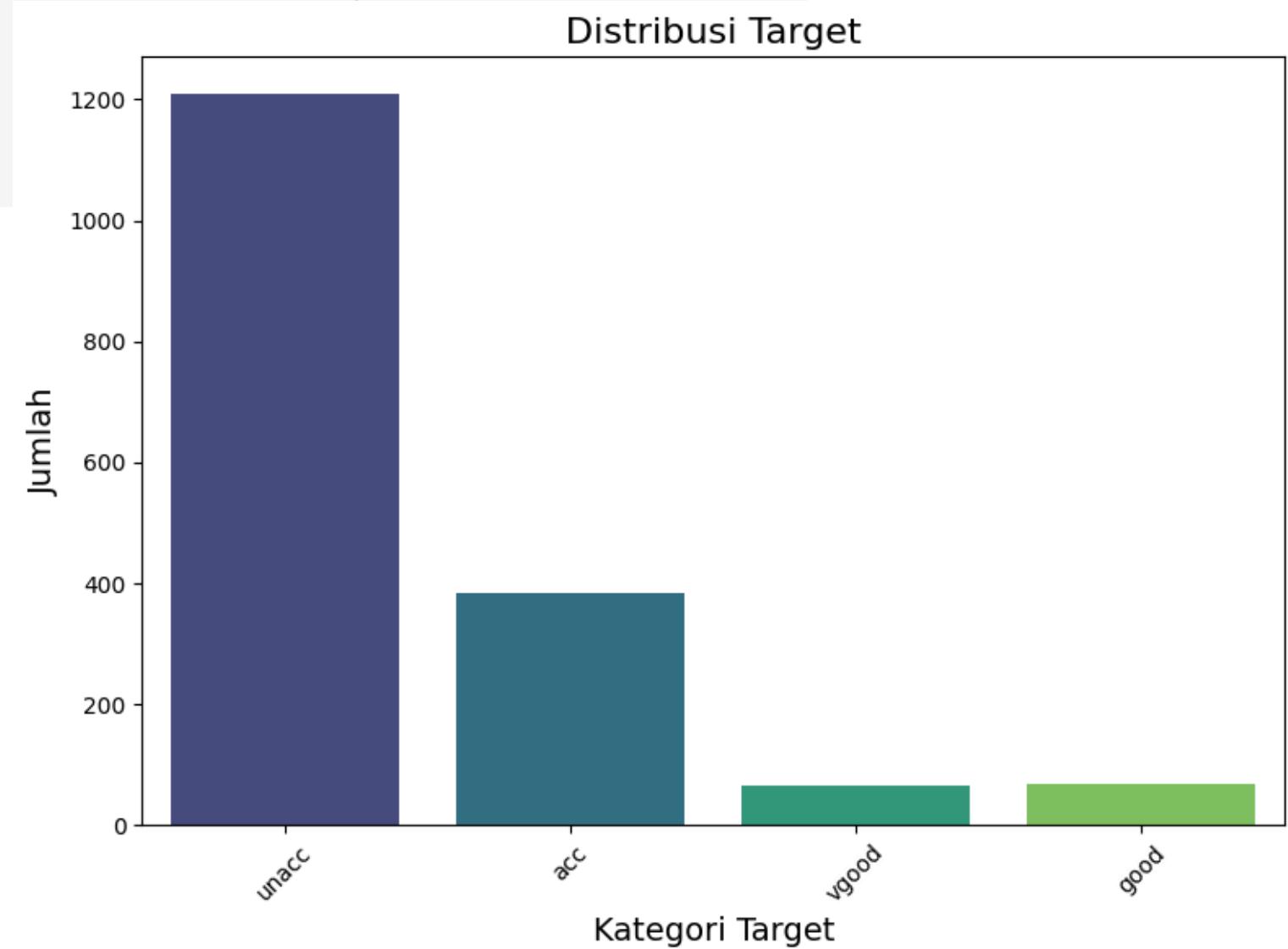
## Data Visualization

```
# Plot distribusi target
plt.figure(figsize=(8, 6))
sns.countplot(data=dataset, x='class', palette='viridis') # Ganti 'class' dengan nama kolom target
plt.title('Distribusi Target', fontsize=16)
plt.xlabel('Kategori Target', fontsize=14)
plt.ylabel('Jumlah', fontsize=14)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

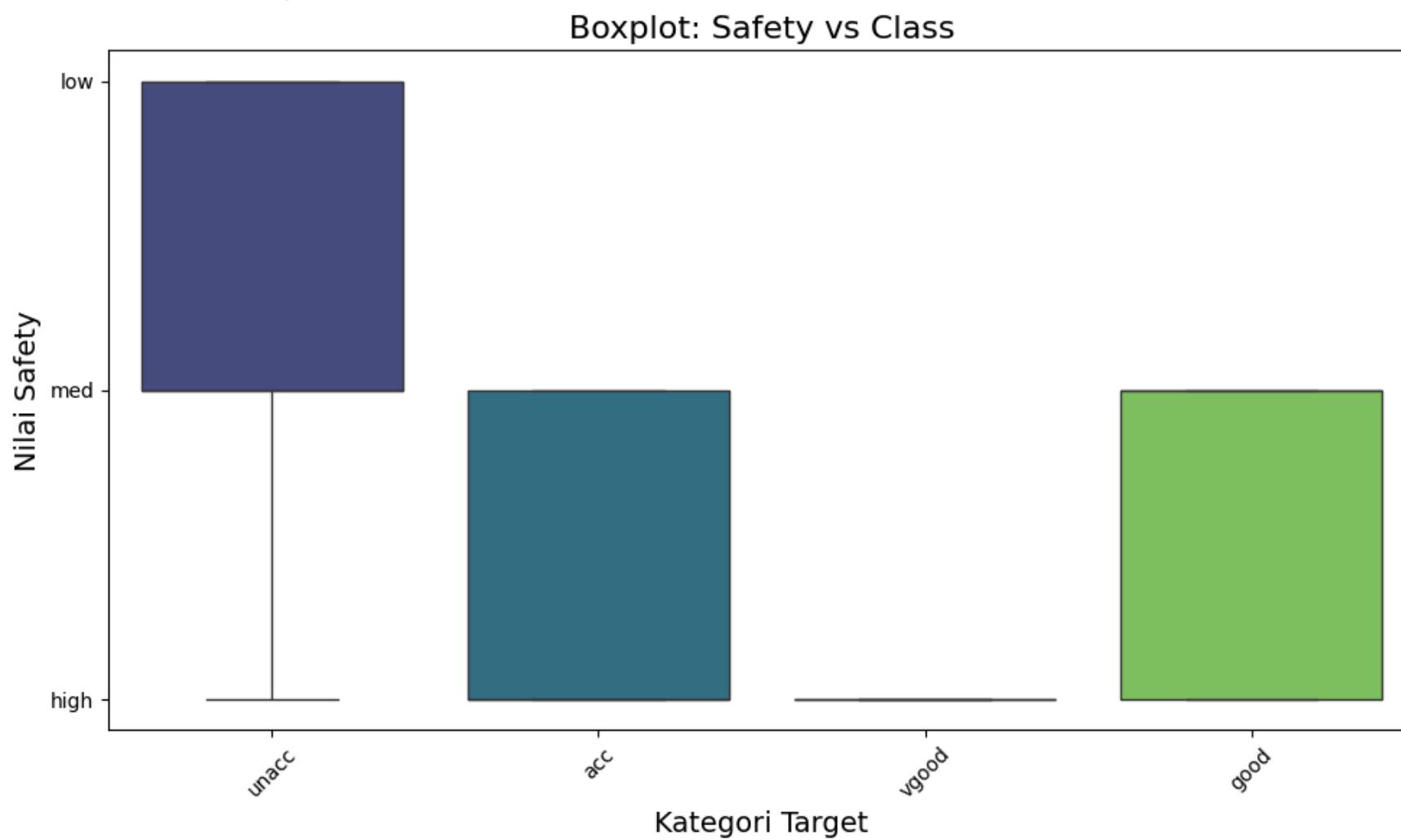
Kategori unacc (tidak diterima) mendominasi dataset dengan jumlah lebih dari 1.200 instance.

Kategori acc (dapat diterima) memiliki jumlah sedang, sekitar 400 instance.

Kategori vgood (sangat baik) dan good (baik) memiliki jumlah instance yang jauh lebih sedikit, masing-masing kurang dari 100 instance.



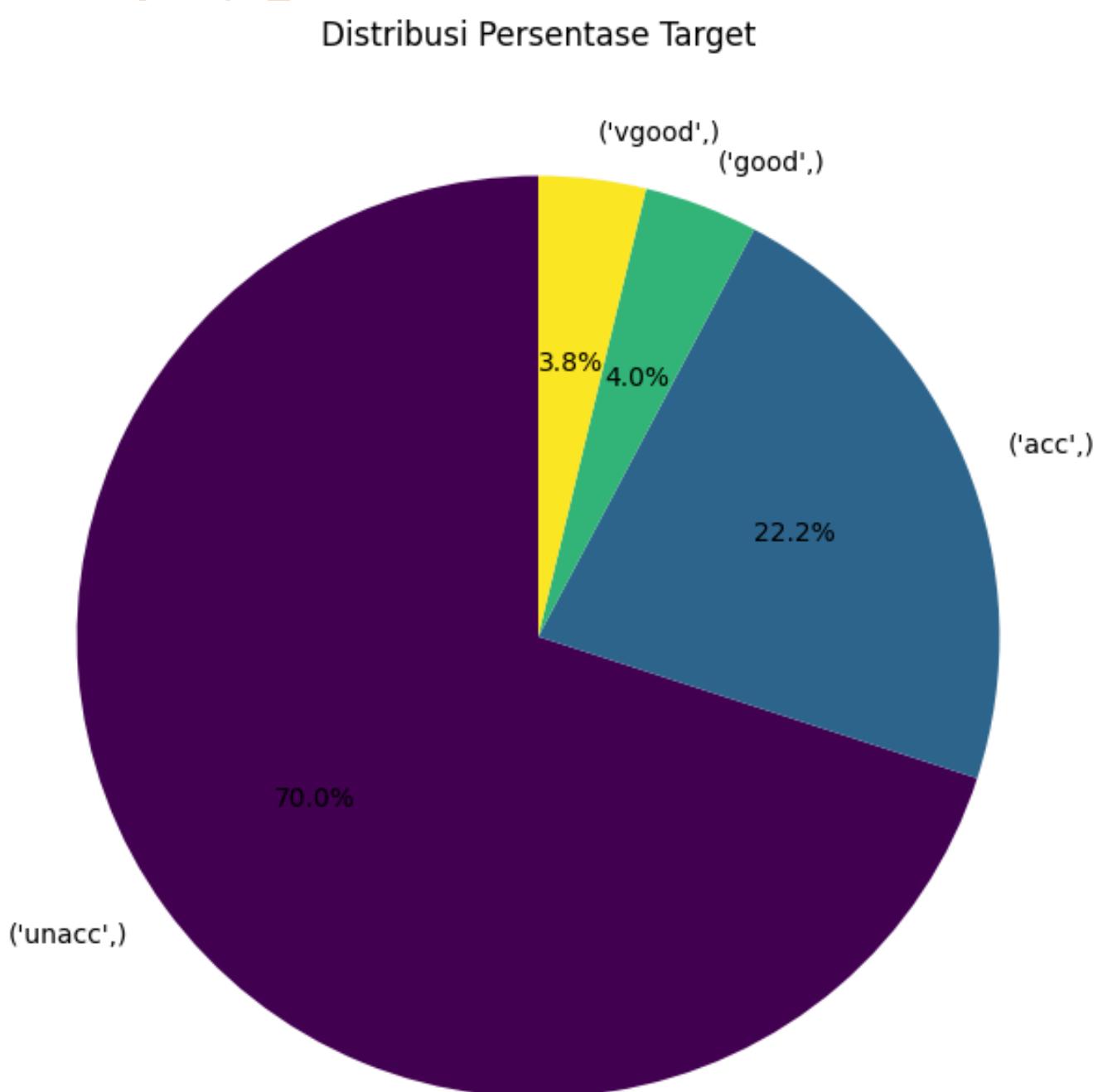
# Box Plot



## Hubungan Safety dengan Kategori Target:

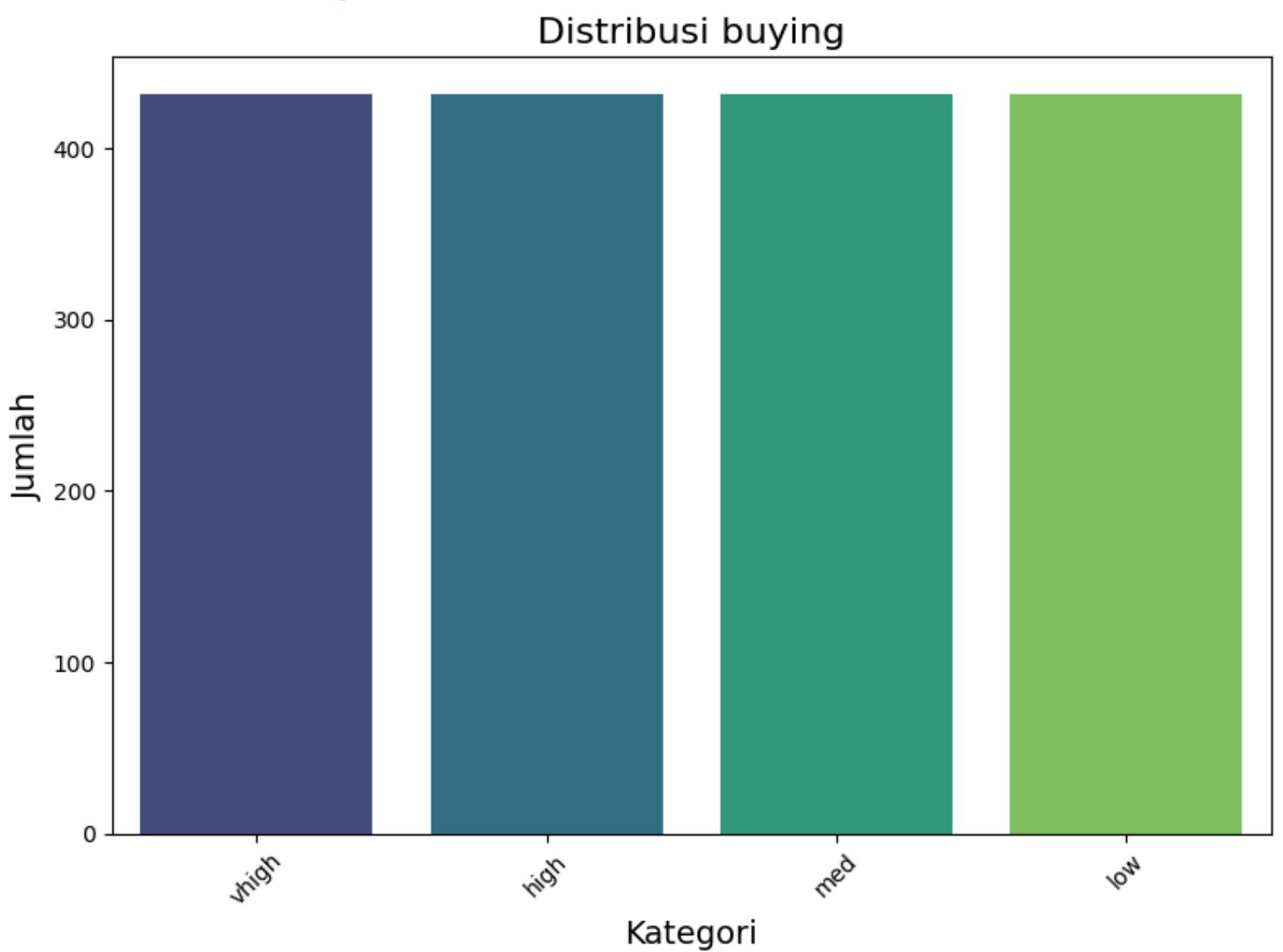
- Nilai safety (keamanan) untuk kategori unacc (tidak dapat diterima) lebih bervariasi, mencakup rentang dari low hingga med, dengan sebagian besar data berada pada nilai yang lebih rendah.
- Untuk kategori acc (dapat diterima), nilai safety umumnya berada di rentang med, menunjukkan tingkat keamanan yang lebih baik dibandingkan unacc.
- Pada kategori good dan vgood, nilai safety cenderung stabil pada med hingga high, yang berarti mobil dalam kategori ini memiliki tingkat keamanan yang lebih tinggi.

# Pie Chart



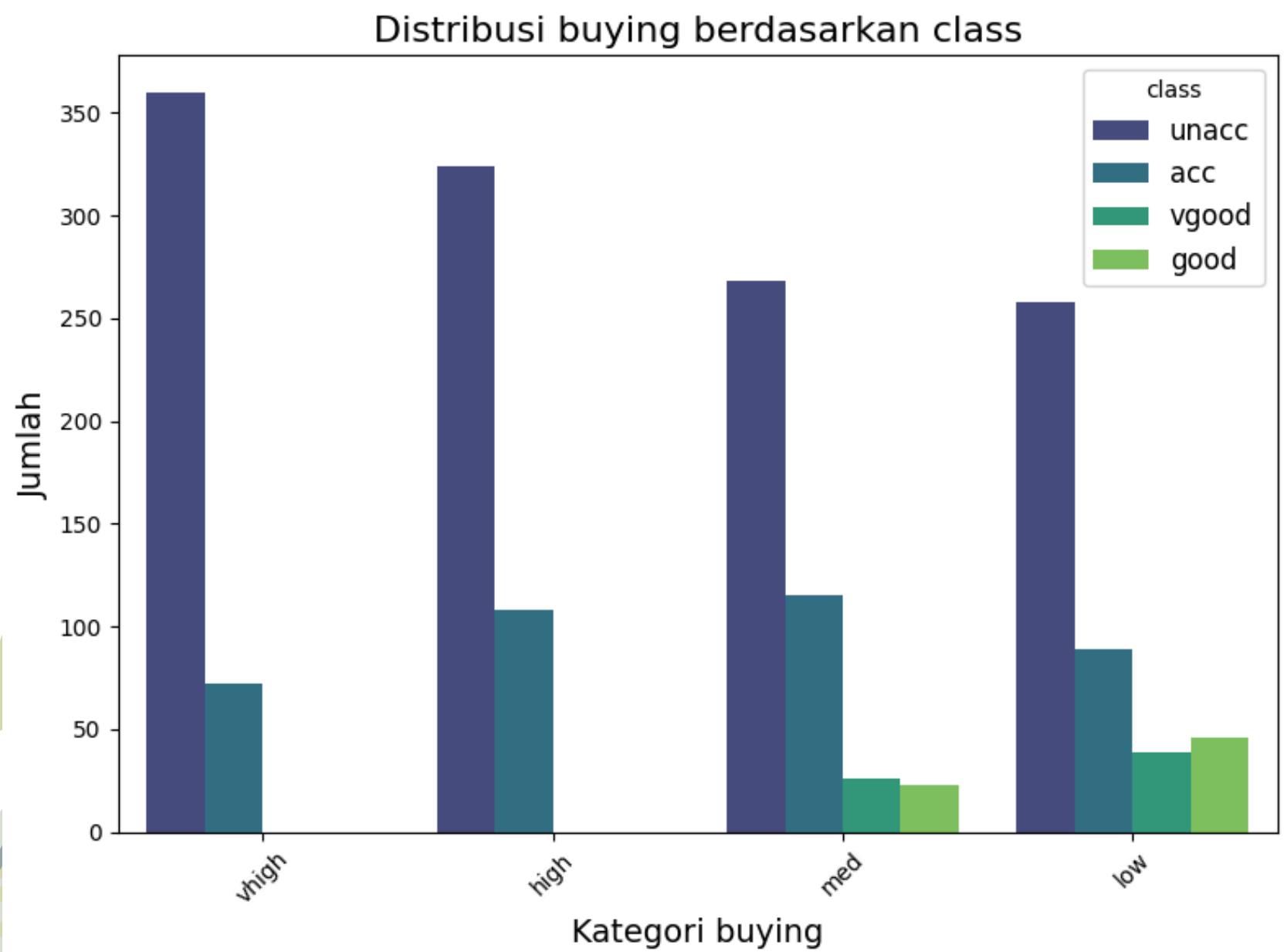
- Kategori acc (dapat diterima) memiliki porsi sebesar 22.2%, menempati posisi kedua terbesar.
- Kategori vgood (sangat baik) dan good (baik) hanya menyumbang 3.8% dan 4.0% dari total data, menjadikannya kategori minoritas.

# Bar Plot



- Semua kategori pada fitur buying (vhigh, high, med, low) memiliki jumlah data yang hampir sama, sekitar 400-450 instance per kategori.
- Hal ini menunjukkan bahwa dataset memiliki distribusi data yang seimbang untuk fitur harga pembelian, sehingga tidak ada bias terhadap kategori tertentu.

# Bar Plot



**Kategori unacc Dominan di Semua Tingkat Harga:**

- Untuk semua kategori harga (vhigh, high, med, low), mayoritas mobil masuk dalam kategori unacc (tidak dapat diterima).
- Hal ini menunjukkan bahwa atribut lain, seperti maint, safety, atau lug\_boot, mungkin lebih berpengaruh dalam menentukan evaluasi mobil sebagai "dapat diterima" atau lebih baik.

# Logistics Regression

# Import Library

```
▶ from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.linear_model import LogisticRegression  
from sklearn.pipeline import Pipeline  
from sklearn.model_selection import GridSearchCV  
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, auc  
from sklearn.preprocessing import label_binarize  
import matplotlib.pyplot as plt  
import seaborn as sns  
import pandas as pd  
import numpy as np
```

# Memisahkan Dataset & Buat Pipeline

```
[ ] # Mengonversi fitur kategorikal menjadi numerik menggunakan One-Hot Encoding  
X_encoded = pd.get_dummies(X)  
  
[ ] # Memisahkan dataset menjadi fitur (X_encoded) dan target (y)  
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)  
  
▶ # Membuat pipeline dengan Logistic Regression  
pipeline = Pipeline([  
    ('scaler', StandardScaler()), # Menstandarisasi fitur  
    ('logistic_regression', LogisticRegression(max_iter=1000)) # Model Logistic Regression  
])
```

# Model Evaluation

```
# GridSearchCV
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy', verbose=1, n_jobs=-1)
grid_search.fit(X_train, y_train)

# Menampilkan hasil terbaik
print("Best parameters (GridSearchCV):", grid_search.best_params_)
print("Best score (GridSearchCV):", grid_search.best_score_)

# Evaluasi pada test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
print("Accuracy on test set (GridSearchCV):", accuracy_score(y_test, y_pred))
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWarning
  y = column_or_1d(y, warn=True)
Best parameters (GridSearchCV): {'C': 100, 'penalty': 'l1', 'solver': 'liblinear'}
Best score (GridSearchCV): 0.8972453304033904
Accuracy on test set (GridSearchCV): 0.8988439306358381
```

---

Model terbaik (`best_model`) diterapkan pada data pengujian, menghasilkan akurasi sebesar 0.8984 (sekitar 89.84%). Ini menunjukkan bahwa performa pada test set hampir sama dengan performa rata-rata pada data pelatihan (cross-validation), yang merupakan tanda model yang baik dan tidak overfitting.

# Classification Report

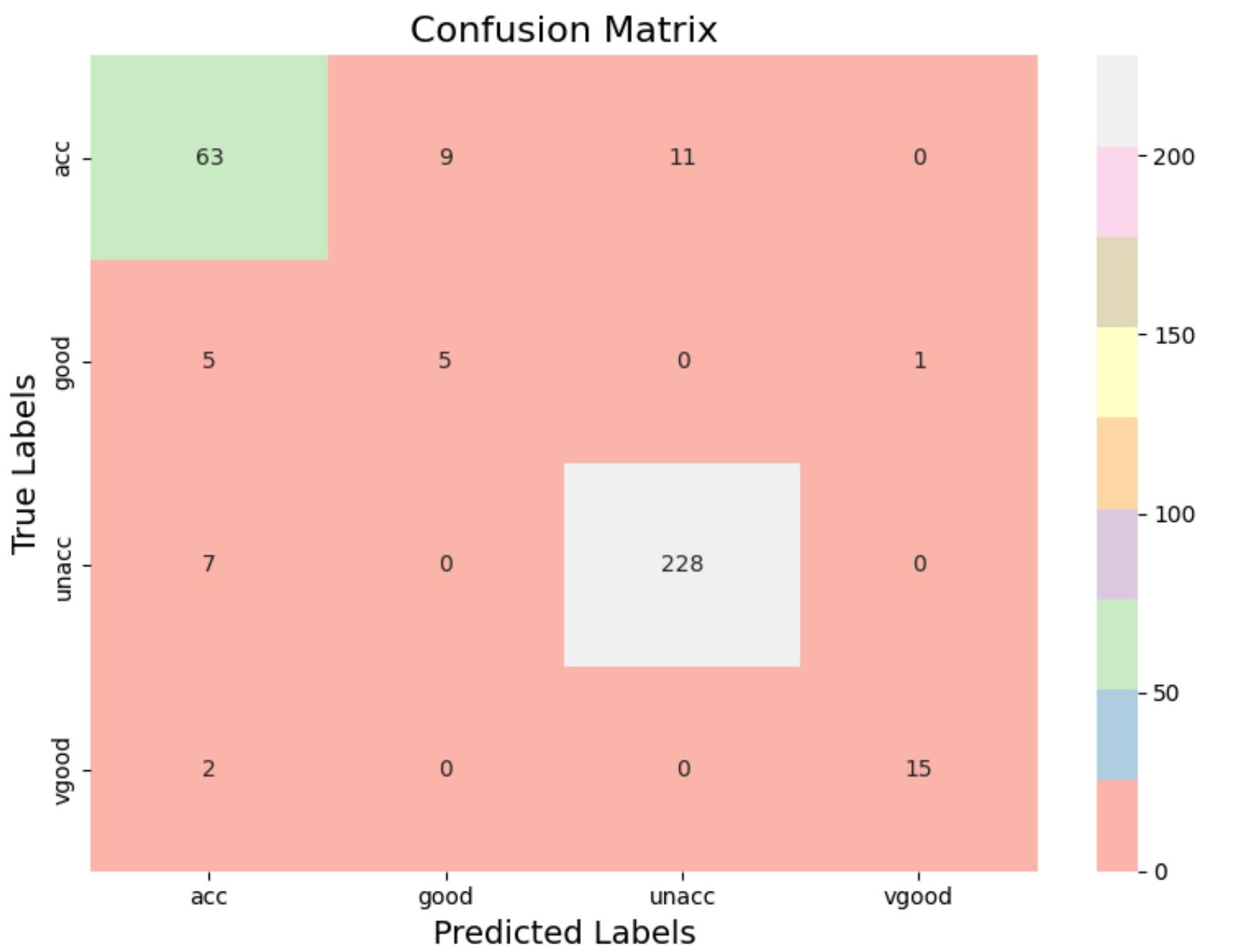
Classification Report:

	precision	recall	f1-score	support
acc	0.82	0.76	0.79	83
good	0.36	0.45	0.40	11
unacc	0.95	0.97	0.96	235
vgood	0.94	0.88	0.91	17
accuracy			0.90	346
macro avg	0.77	0.77	0.76	346
weighted avg	0.90	0.90	0.90	346

## Kinerja Keseluruhan:

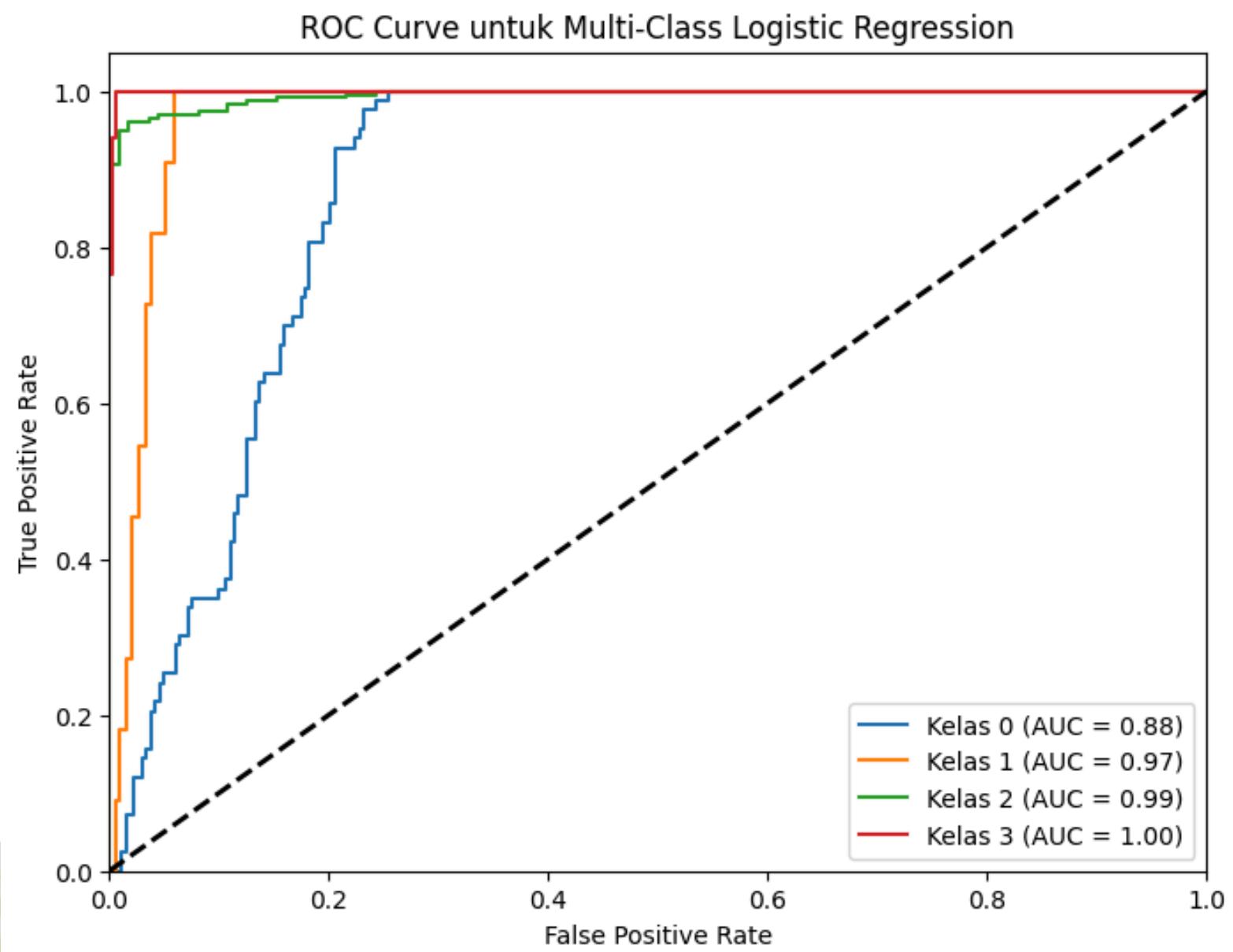
- **Accuracy:** Model memiliki akurasi keseluruhan sebesar 90%, yang menunjukkan performa yang sangat baik dalam memprediksi target.
- **Weighted Avg:** Precision, recall, dan F1-score semuanya berada di 90%, yang sesuai dengan distribusi data dan mendukung hasil akurasi yang tinggi.

# Heatmap



**True Positives:** Model berhasil memprediksi 228 instance kelas unacc dengan benar.

# ROC Curve



- Kelas 0 memiliki nilai AUC yang lebih rendah dibandingkan kelas lainnya. Ini mengindikasikan bahwa kelas ini lebih sulit dibedakan dari kelas lain oleh model.
- Kelas 2 dan 3 memiliki nilai AUC mendekati atau sama dengan 1, menunjukkan bahwa model sangat akurat dalam memprediksi kedua kelas tersebut.

# Decision Tree

# Model Evaluation

```
▶ # Menampilkan hasil terbaik dari GridSearchCV  
print("Best parameters (GridSearchCV):", grid_search.best_params_)  
print("Best score (GridSearchCV):", grid_search.best_score_)  
  
# Evaluasi pada test set  
best_model_grid = grid_search.best_estimator_  
y_pred_grid = best_model_grid.predict(X_test)  
print("Accuracy on test set (GridSearchCV):", accuracy_score(y_test, y_pred_grid))
```

→ Best parameters (GridSearchCV): {'decision\_tree\_criterion': 'log\_loss', 'decision\_}  
Best score (GridSearchCV): 0.9674253126144509  
Accuracy on test set (GridSearchCV): 0.9653179190751445

- **Accuracy on Test Set: 0.9653 (96.53%) menunjukkan bahwa model berhasil mempertahankan performa tinggi pada data pengujian.**
- **Performa pada data test hampir sama dengan data training (GridSearchCV), yang merupakan indikasi bahwa model ini memiliki generalisasi yang baik dan tidak overfitting.**

# Classification Report

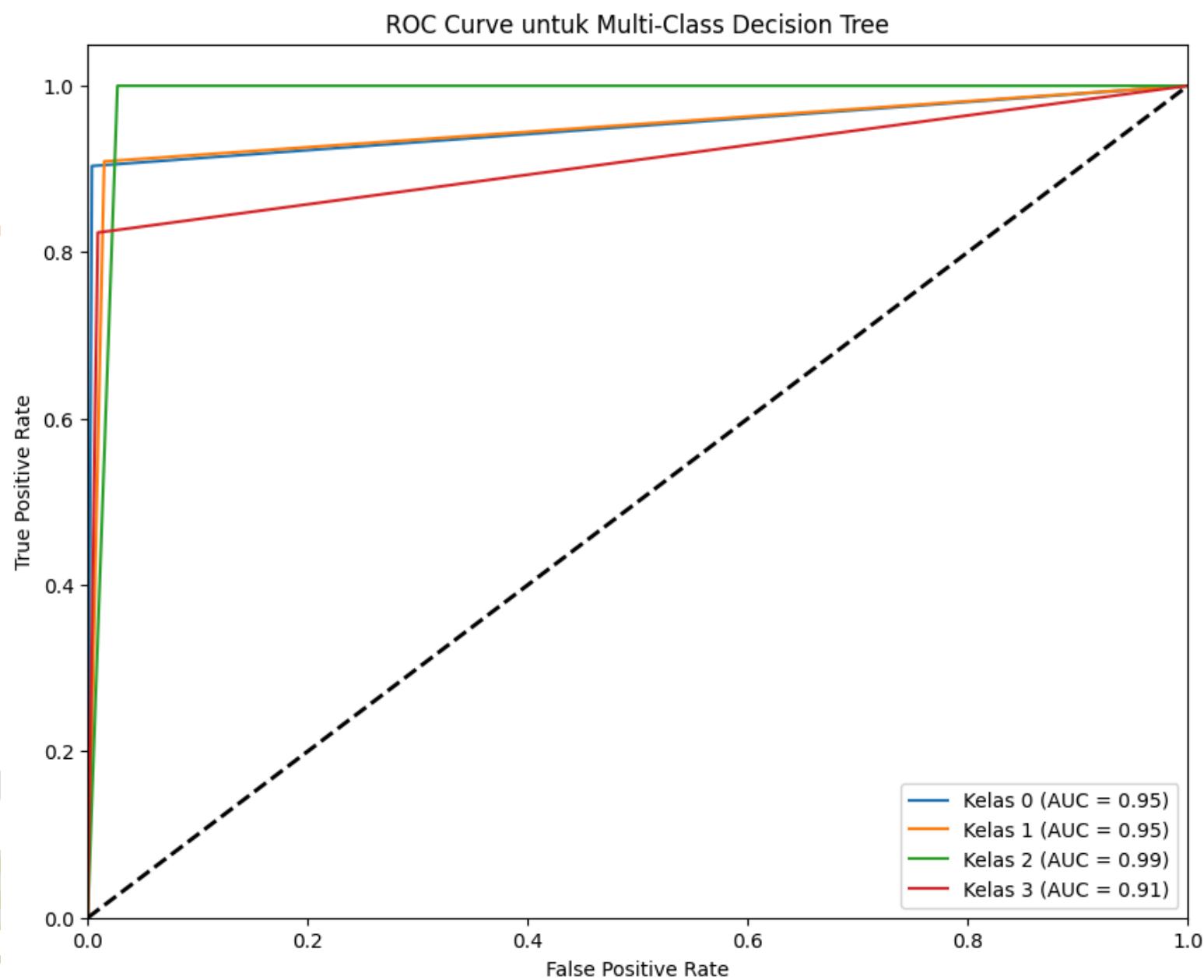
→ Classification Report:

	precision	recall	f1-score	support
acc	0.82	0.76	0.79	83
good	0.36	0.45	0.40	11
unacc	0.95	0.97	0.96	235
vgood	0.94	0.88	0.91	17
accuracy			0.90	346
macro avg	0.77	0.77	0.76	346
weighted avg	0.90	0.90	0.90	346

**Akurasi:** Model memiliki akurasi keseluruhan sebesar 90%, menunjukkan performa yang sangat baik dalam memprediksi target.

**Weighted Average:** Precision, recall, dan F1-score masing-masing adalah 90%. Ini menunjukkan bahwa model mempertimbangkan distribusi kelas secara seimbang dalam metrik keseluruhan.

# ROC Curve



- Model Decision Tree memiliki performa yang sangat baik untuk sebagian besar kelas dengan AUC di atas 0.95.
- Performa pada kelas 2 ( $AUC = 0.99$ ) adalah yang terbaik, sementara kelas 3 ( $AUC = 0.91$ ) menunjukkan peluang untuk perbaikan, terutama dengan menangani ketidakseimbangan data.

# K-Nearest Neighbors

# Model Evaluation

```
▶ # Menampilkan hasil terbaik dari GridSearchCV  
print("Best parameters (GridSearchCV):", grid_search.best_params_)  
print("Best score (GridSearchCV):", grid_search.best_score_)  
  
# Evaluasi pada test set  
best_model_grid = grid_search.best_estimator_  
y_pred_grid = best_model_grid.predict(X_test)  
print("Accuracy on test set (GridSearchCV):", accuracy_score(y_test, y_pred_grid))
```

→ Best parameters (GridSearchCV): {'metric': 'manhattan', 'n\_neighbors': 11, 'weights': 'distance'}  
Best score (GridSearchCV): 0.9218463872756761  
Accuracy on test set (GridSearchCV): 0.9190751445086706

- Accuracy on Test Set: 0.9191 (91.91%) menunjukkan bahwa model mampu mempertahankan performa tinggi pada data pengujian.
- Performa pada test set sangat mendekati hasil pada data pelatihan (cross-validation), menunjukkan bahwa model ini memiliki generalisasi yang baik tanpa overfitting.

# Classification Report

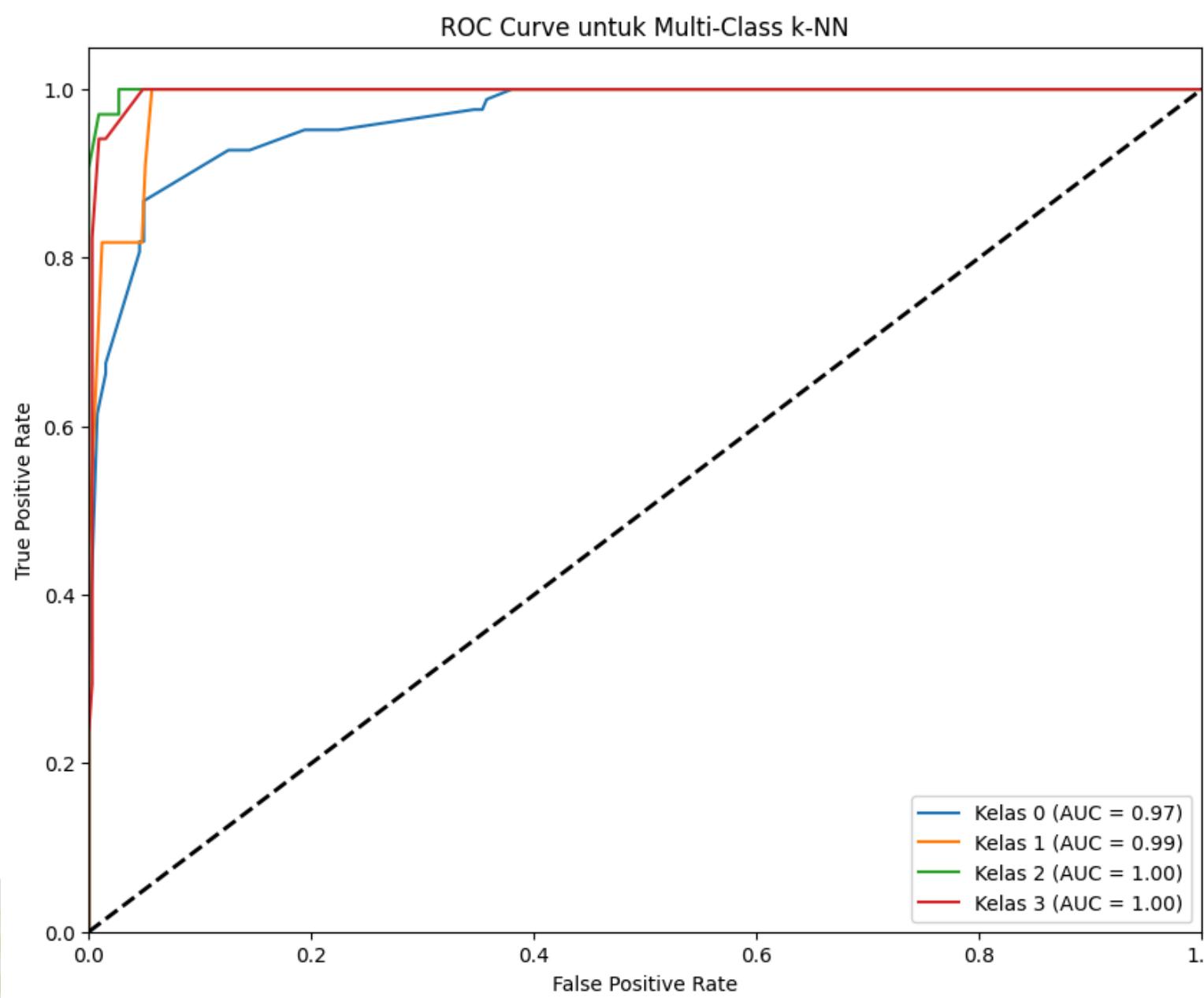


Classification Report:

	precision	recall	f1-score	support
acc	0.82	0.76	0.79	83
good	0.36	0.45	0.40	11
unacc	0.95	0.97	0.96	235
vgood	0.94	0.88	0.91	17
accuracy			0.90	346
macro avg	0.77	0.77	0.76	346
weighted avg	0.90	0.90	0.90	346

- **Akurasi:** Model memiliki akurasi keseluruhan sebesar 90%, menunjukkan bahwa model bekerja sangat baik untuk dataset ini.
- **Macro Average:** Precision, recall, dan F1-score masing-masing sekitar 0.77.
- **Weighted Average:** Precision, recall, dan F1-score berada di 0.90, mencerminkan bahwa metrik keseluruhan sangat dipengaruhi oleh performa model pada kelas mayoritas (unacc), yang memiliki jumlah data paling banyak.

# ROC Curve



- Model k-NN memiliki performa luar biasa dalam membedakan kelas target, dengan semua AUC mendekati atau sama dengan 1.
- Meskipun kelas 0 sedikit lebih sulit untuk dibedakan dibandingkan kelas lainnya, performanya tetap sangat baik.

# XGBoost Classification

# Model Evaluation

```
▶ # Menampilkan hasil terbaik dari GridSearchCV  
print("Best parameters (GridSearchCV):", grid_search.best_params_)  
print("Best score (GridSearchCV):", grid_search.best_score_)  
  
# Evaluasi pada test set  
best_model_grid = grid_search.best_estimator_  
y_pred_grid = best_model_grid.predict(X_test)  
print("Accuracy on test set (GridSearchCV):", accuracy_score(y_test, y_pred_grid))
```

```
→ Best parameters (GridSearchCV): {'xgb__colsample_bytree': 1.0, 'xgb__gamma': 0, 'xgb__le  
Best score (GridSearchCV): 0.9891435148851567  
Accuracy on test set (GridSearchCV): 0.976878612716763
```

**Best Score (GridSearchCV): 0.9891 (98.91%) menunjukkan bahwa model memiliki performa sangat baik pada data pelatihan dengan validasi silang.**

# Classification Report

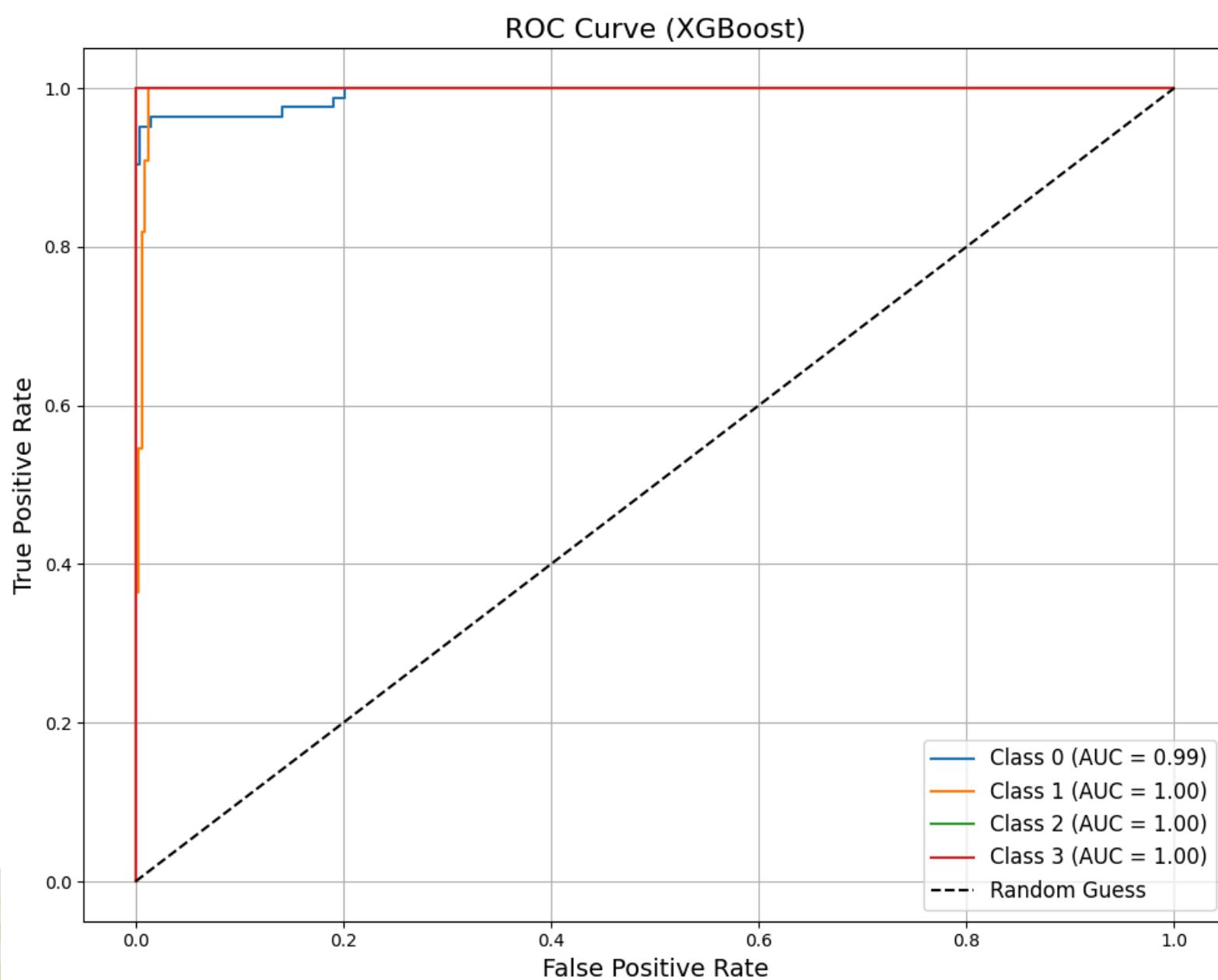


Classification Report:

	precision	recall	f1-score	support
acc	0.82	0.76	0.79	83
good	0.36	0.45	0.40	11
unacc	0.95	0.97	0.96	235
vgood	0.94	0.88	0.91	17
accuracy			0.90	346
macro avg	0.77	0.77	0.76	346
weighted avg	0.90	0.90	0.90	346

- **Akurasi:** Model memiliki akurasi keseluruhan sebesar 90%, menunjukkan bahwa model bekerja sangat baik untuk dataset ini.
- **Macro Average:** Precision, recall, dan F1-score masing-masing sekitar 0.77.
- **Weighted Average:** Precision, recall, dan F1-score berada di 0.90, mencerminkan bahwa metrik keseluruhan sangat dipengaruhi oleh performa model pada kelas mayoritas (unacc), yang memiliki jumlah data paling banyak.

# ROC Curve



- Kelas 0 memiliki AUC sedikit lebih rendah dibandingkan kelas lainnya ( $AUC = 0.99$ ), tetapi tetap berada dalam kategori sangat baik.
- Hal ini menunjukkan bahwa meskipun model memiliki performa tinggi untuk kelas ini, masih terdapat sedikit ruang untuk perbaikan dalam membedakan kelas ini dari yang lain.