Tugas Kelompok Berbasis Kasus 01

Nama kelompok : Kelompok Kosong

Anggota Kelompok

- Tresnawan 122140178

- Ferdinand Yehezkiel Hutapea 122140233

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
warnings.filterwarnings('ignore')

# Set style untuk visualisasi
sns.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (10, 6)
```

```
df_iris = pd.read_csv('IRIS.csv')

print("\nDataset IRIS berhasil dimuat!")
print(f"Ukuran dataset: {df_iris.shape}")
print("\n5 Baris Pertama:")
print(df_iris.head())
print("\nInformasi Dataset:")
print(df_iris.info())
print("\nKolom yang tersedia:")
print(df_iris.columns.tolist())

# Deteksi nama kolom target (bisa 'species', 'Species', atau lainnya)
target_col = None
for col in df_iris.columns:
    if col.lower() in ['species', 'class', 'target', 'label']:
        target_col = col
        break

if target_col is None:
    target_col = df_iris.columns[-1]

print(f"\n Kolom target yang digunakan: '{target_col}'")
print(f" Distribusi Kelas Target:")
print(df_iris[target_col].value_counts())
```

```

Dataset IRIS berhasil dimuat!
Ukuran dataset: (150, 5)

5 Baris Pertama:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

Kolom yang tersedia:
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

```
 Kolom target yang digunakan: 'species'
 Distribusi Kelas Target:
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

```
X = df_iris.drop(target_col, axis=1)
y = df_iris[target_col]

# Hapus kolom ID jika ada
if 'Id' in X.columns or 'id' in X.columns or 'ID' in X.columns:
    id_col = [col for col in X.columns if col.lower() == 'id'][0]
    X = X.drop(id_col, axis=1)
    print(f"\n Kolom '{id_col}' dihapus dari fitur")

print("\nFitur (X) dan Target (y) berhasil dipisahkan")
print(f"Jumlah fitur: {X.shape[1]}")
print(f"Nama fitur: {X.columns.tolist()}")
print(f"Jumlah sampel: {X.shape[0]}")
```

```
Fitur (X) dan Target (y) berhasil dipisahkan
Jumlah fitur: 4
Nama fitur: ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
Jumlah sampel: 150
```

```
le = LabelEncoder()
y_encoded = le.fit_transform(y)

print("\nLabel Encoding:")
for i, class_name in enumerate(le.classes_):
    print(f"  {class_name} → {i}")
```

```
Label Encoding:
  Iris-setosa → 0
  Iris-versicolor → 1
  Iris-virginica → 2
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("\nFitur berhasil dinormalisasi dengan StandardScaler")
print("Statistik sebelum normalisasi:")
print(X.describe().loc[['mean', 'std']].round(3))
print("\nStatistik setelah normalisasi:")
print(pd.DataFrame(X_scaled, columns=X.columns).describe().loc[['mean', 'std']].round(3))
```

```
Fitur berhasil dinormalisasi dengan StandardScaler
Statistik sebelum normalisasi:
      sepal_length  sepal_width  petal_length  petal_width
mean         5.843        3.054         3.759        1.199
std          0.828        0.434         1.764        0.763

Statistik setelah normalisasi:
      sepal_length  sepal_width  petal_length  petal_width
mean        -0.000       -0.000         0.000       -0.000
std          1.003        1.003         1.003        1.003
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded,
    test_size=0.2,
    random_state=42,
    stratify=y_encoded
)

print(f"\nData Split:")
print(f"  Training set: {X_train.shape[0]} sampel")
print(f"  Testing set: {X_test.shape[0]} sampel")
```

```
Data Split:
  Training set: 120 sampel
  Testing set: 30 sampel
```

```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1],
```

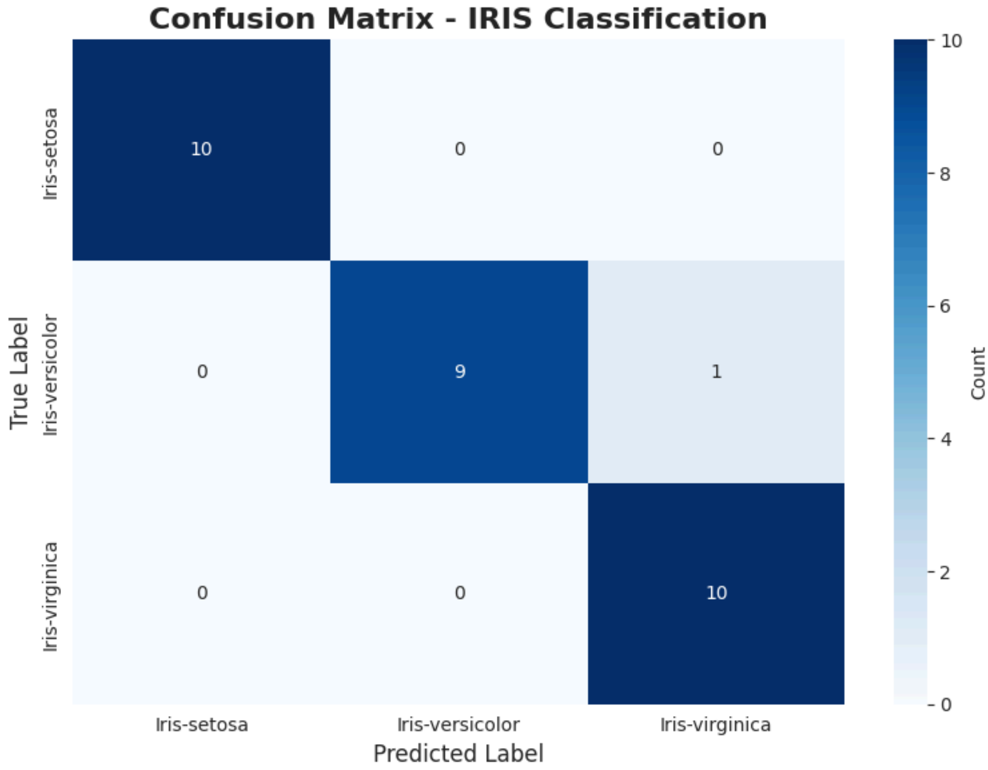
```
    'kernel': ['rbf', 'poly', 'sigmoid']
}

# GridSearchCV
grid_search = GridSearchCV(
    SVC(random_state=42),
    param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

print("\nMencari parameter terbaik...")
grid_search.fit(X_train, y_train)

print(f"\nParameter Terbaik: {grid_search.best_params_}")
print(f"Cross-Validation Score Terbaik: {grid_search.best_score_:.4f}")

# Model terbaik
best_svc = grid_search.best_estimator_
```

```
Mencari parameter terbaik...
Fitting 5 folds for each of 72 candidates, totalling 360 fits

Parameter Terbaik: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
Cross-Validation Score Terbaik: 0.9833
```

```
y_pred = best_svc.predict(X_test)

# Akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAkurasi Model: {accuracy * 100:.2f}%")

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)
```

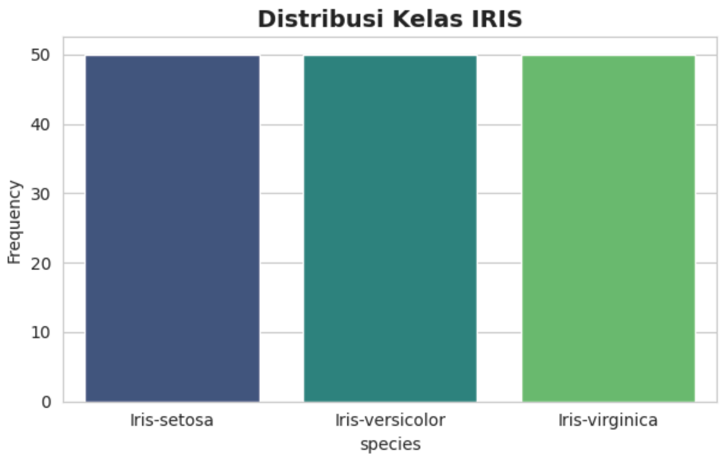
```
Akurasi Model: 96.67%

Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      0.90      0.95        10
 Iris-virginica       0.91      1.00      0.95        10

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30


Confusion Matrix:
[[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
```

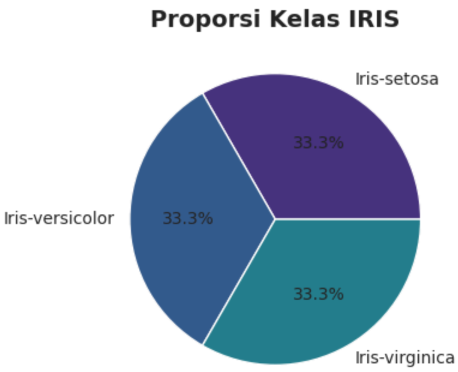
```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=le.classes_,
            yticklabels=le.classes_,
            cbar_kws={'label': 'Count'})
plt.title('Confusion Matrix - IRIS Classification', fontsize=16, fontweight='bold')
plt.ylabel('True Label', fontsize=12)
plt.xlabel('Predicted Label', fontsize=12)
plt.tight_layout()
plt.show()
```




```python
print("\nStatistik Deskriptif:")
print(df_iris.describe())
print("\nInformasi Dataset:")
print(df_iris.info())
print("\nMissing Values:")
print(df_iris.isnull().sum())
print("\nNama Kolom:")
print(df_iris.columns.tolist())

# Deteksi nama kolom target (species)
target_col_iris = None
for col in df_iris.columns:
    if col.lower() in ['species', 'class', 'target', 'label']:
        target_col_iris = col
        break

if target_col_iris is None:
    target_col_iris = df_iris.columns[-1]

print(f"\nKolom target yang digunakan: '{target_col_iris}'")

# Visualisasi distribusi target
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
sns.countplot(data=df_iris, x=target_col_iris, palette='viridis')
plt.title('Distribusi Kelas IRIS', fontsize=14, fontweight='bold')
plt.xlabel(f'{target_col_iris}')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
df_iris[target_col_iris].value_counts().plot(kind='pie', autopct='%1.1f%%', colors=sns.color_palette('viridis'))
plt.title('Proporsi Kelas IRIS', fontsize=14, fontweight='bold')
plt.ylabel('')

plt.tight_layout()
plt.show()
```

```
Statistik Deskriptif:
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

Missing Values:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64

Nama Kolom:
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']

Kolom target yang digunakan: 'species'
```






```
df_processed = df_iris.copy()

target_col_iris = None
for col in df_iris.columns:
    if col.lower() in ['species', 'class', 'target', 'label']:
        target_col_iris = col
        break

if target_col_iris is None:
    target_col_iris = df_iris.columns[-1]

# Identifikasi kolom kategorikal (excluding the target column)
categorical_cols = df_processed.select_dtypes(include=['object']).columns.tolist()
if target_col_iris in categorical_cols:
    categorical_cols.remove(target_col_iris)


print(f"\nKolom kategorikal yang terdeteksi (selain target): {categorical_cols}")

# For the Iris dataset, there are no other categorical columns to encode
if not categorical_cols:
    print("\nTidak ada kolom kategorikal lain yang perlu di-encode di dataset IRIS.")
```

```
encoded_cols = []
onehot_cols = []

for col in categorical_cols:
    unique_values = df_processed[col].nunique()

    if unique_values == 2:
        le_temp = LabelEncoder()
        df_processed[col] = le_temp.fit_transform(df_processed[col])
        encoded_cols.append(col)
        print(f"\nLabel Encoding '{col}': {dict(zip(le_temp.classes_, le_temp.transform(le_temp.classes_)))}")
    elif unique_values > 2:
        onehot_cols.append(col)

if onehot_cols:
    df_processed = pd.get_dummies(df_processed, columns=onehot_cols, drop_first=True)
    new_cols = [col for col in df_processed.columns if any(oh in col for oh in onehot_cols)]
    print(f"\nOne-Hot Encoding untuk {onehot_cols}")
    print(f"  Kolom baru: {new_cols}")

print(f"\nUkuran dataset setelah (attempted) encoding: {df_processed.shape}")
print("\nPreview data setelah (attempted) encoding:")
print(df_processed.head())
```

```
Kolom kategorikal yang terdeteksi (selain target): []

Tidak ada kolom kategorikal lain yang perlu di-encode di dataset IRIS.

Ukuran dataset setelah (attempted) encoding: (150, 5)

Preview data setelah (attempted) encoding:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
```

```
df_processed_iris = df_iris.copy()

target_col_iris = None
for col in df_processed_iris.columns:
    if col.lower() in ['species', 'class', 'target', 'label']:
        target_col_iris = col
        break

if target_col_iris is None:
    target_col_iris = df_processed_iris.columns[-1]

X_iris = df_processed_iris.drop(target_col_iris, axis=1)
y_iris = df_processed_iris[target_col_iris]

print("\nFeatures (X_iris) and Target (y_iris) separated.")
print(f"Shape of X_iris: {X_iris.shape}")
print(f"Shape of y_iris: {y_iris.shape}")

scaler_iris = StandardScaler()
X_iris_scaled = scaler_iris.fit_transform(X_iris)

print("\nFeatures normalized with StandardScaler.")
print("Shape of scaled features (X_iris_scaled):", X_iris_scaled.shape)

print(f"\nJumlah fitur: {X_iris.shape[1]}")
print(f"Fitur yang digunakan: {list(X_iris.columns)}")
```

```
Features (X_iris) and Target (y_iris) separated.
Shape of X_iris: (150, 4)
Shape of y_iris: (150,)

Features normalized with StandardScaler.
Shape of scaled features (X_iris_scaled): (150, 4)

Jumlah fitur: 4
Fitur yang digunakan: ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled,
    y_encoded,
    test_size=0.2,
    random_state=42,
```

```
    stratify=y_encoded
)

print(f"\nData Split (80:20):")
print(f"  Training set: {X_train.shape[0]} samples")
print(f"  Testing set: {X_test.shape[0]} samples")
```

```
Data Split (80:20):
  Training set: 120 samples
  Testing set: 30 samples
```

```
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression

print("\nMelatih SVR pada data Iris")
param_grid_svr = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.001, 0.01],
    'kernel': ['rbf', 'linear']
}

grid_svr = GridSearchCV(
    SVR(),
    param_grid_svr,
    cv=5,
    scoring='r2',
    n_jobs=-1,
    verbose=1
)

grid_svr.fit(X_train, y_train)
best_svr = grid_svr.best_estimator_

print(f"\nParameter SVR Terbaik (berdasarkan skor R² pada data klasifikasi): {grid_svr.best_params_}")
print(f"Skor Cross-Validation R² Terbaik: {grid_svr.best_score_:.4f}")
print("\nModel SVR berhasil dilatih.")

lr_model = LinearRegression()

lr_model.fit(X_train, y_train)
print("Model Linear Regression berhasil dilatih.")
```

```
Melatih SVR pada data Iris
Fitting 5 folds for each of 32 candidates, totalling 160 fits

Parameter SVR Terbaik (berdasarkan skor R² pada data klasifikasi): {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
Skor Cross-Validation R² Terbaik: 0.9415

Model SVR berhasil dilatih.
Model Linear Regression berhasil dilatih.
```

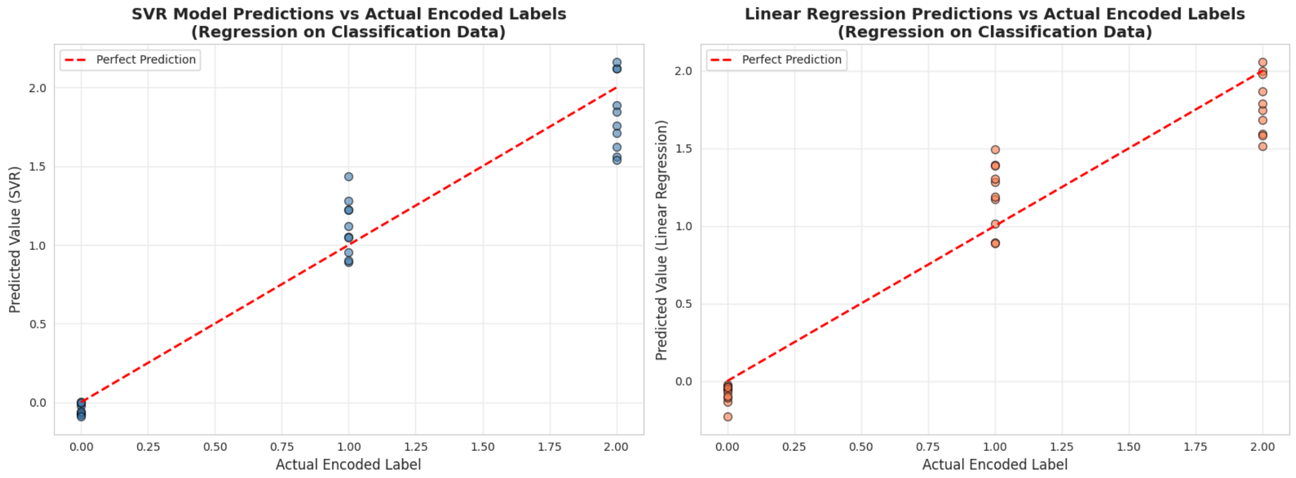
```
Mulai coding atau buat kode dengan AI.
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

y_pred_svr = best_svr.predict(X_test)
y_pred_lr = lr_model.predict(X_test)

y_test_encoded = y_test
def evaluate_regression(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_true, y_pred)

    print(f"Evaluasi Regresi untuk {model_name}")

    print(f"  MAE  (Mean Absolute Error)    : {mae:,.4f}")
    print(f"  MSE  (Mean Squared Error)     : {mse:,.4f}")
    print(f"  RMSE (Root Mean Squared Error): {rmse:,.4f}")
    print(f"  R²   (R-Squared Score)        : {r2:.4f}")

    return {'MAE': mae, 'MSE': mse, 'RMSE': rmse, 'R2': r2}

metrics_svr = evaluate_regression(y_test_encoded, y_pred_svr, "Support Vector Regressor (SVR)")
metrics_lr = evaluate_regression(y_test_encoded, y_pred_lr, "Linear Regression")
```

```
Evaluasi Regresi untuk Support Vector Regressor (SVR)
  MAE  (Mean Absolute Error)    : 0.1531
```

```
  MSE  (Mean Squared Error)      : 0.0411
  RMSE (Root Mean Squared Error): 0.2028
  R²   (R-Squared Score)         : 0.9383
Evaluasi Regresi untuk Linear Regression
  MAE  (Mean Absolute Error)     : 0.1865
  MSE  (Mean Squared Error)      : 0.0573
  RMSE (Root Mean Squared Error): 0.2395
  R²   (R-Squared Score)         : 0.9140
```

```
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

axes[0].scatter(y_test_encoded, y_pred_svr, alpha=0.6, color='steelblue', edgecolors='black', s=50)

axes[0].plot([y_test_encoded.min(), y_test_encoded.max()],
             [y_test_encoded.min(), y_test_encoded.max()],
             'r--', lw=2, label='Perfect Prediction')
axes[0].set_xlabel('Actual Encoded Label', fontsize=12)
axes[0].set_ylabel('Predicted Value (SVR)', fontsize=12)
axes[0].set_title(f'SVR Model Predictions vs Actual Encoded Labels\n(Regression on Classification Data)',
                  fontsize=14, fontweight='bold')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

axes[1].scatter(y_test_encoded, y_pred_lr, alpha=0.6, color='coral', edgecolors='black', s=50)

axes[1].plot([y_test_encoded.min(), y_test_encoded.max()],
             [y_test_encoded.min(), y_test_encoded.max()],
             'r--', lw=2, label='Perfect Prediction')
axes[1].set_xlabel('Actual Encoded Label', fontsize=12)
axes[1].set_ylabel('Predicted Value (Linear Regression)', fontsize=12)
axes[1].set_title(f'Linear Regression Predictions vs Actual Encoded Labels\n(Regression on Classification Data)',
                  fontsize=14, fontweight='bold')
axes[1].legend()
axes[1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



ANALISIS

Dataset IRIS.csv berhasil dimuat dengan baik ke dalam DataFrame dan terdiri dari 150 sampel dengan 5 kolom, yaitu empat fitur numerik (sepal_length, sepal_width, petal_length, dan petal_width) serta satu kolom target kategorikal (species). Berdasarkan hasil eksplorasi, seluruh kolom memiliki 150 nilai tanpa adanya missing values, sehingga dataset dapat dikatakan bersih dan siap digunakan. Kolom target species berisi tiga kelas yang seimbang — Iris-setosa, Iris-versicolor, dan Iris-virginica — masing-masing berjumlah 50 sampel. Statistik deskriptif menunjukkan sebaran data yang normal tanpa indikasi nilai ekstrem yang signifikan, sedangkan visualisasi seperti count plot dan pie chart menegaskan keseimbangan distribusi antar kelas. Secara keseluruhan, dataset IRIS memiliki kualitas yang baik, seimbang, dan ideal untuk digunakan dalam pelatihan serta pengujian model klasifikasi.