Nama: Andi Cleopatra Maryam Jamila

Nim: 1103213071

Classification model

```
# Install dan impor library
!pip install xgboost
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.preprocessing import StandardScaler, LabelEncoder
from sklearn.pipeline import Pipeline
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
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.1.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.26.4)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.10/dist-packages (from xgboost) (2.23.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.13.1)
```

Memasukkan library yang dibutuhkan dalam Classification model

Memuat Dataset

```
[81] # Menentukan kolom dengan nama yang sesuai
     columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
```

```
[82] # Memuat dataset dengan header yang benar
     df = pd.read_csv("/content/sample_data/iris.data", header=None, names=columns)

     # Menampilkan lima data pertama
     df.head()
```

	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

Menentukan kolom dengan nama yang sesuai, kemudian memasukkan dataset untuk dibaca, dan menampilkan 5 data pertama

```
[83] # Cek nama kolom dan data
     print(df.columns)
     print(df.head())
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
   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
```

Penjelasan: Data iris.csv berisi fitur sepal dan petal beserta label kelas spesies.

Mengecek nama kolom dan data

** Explanatory Data Analysis (EDA)**

a) Statistik Deskriptif

```
[84] # Statistik deskriptif
     df.describe()
```

	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

Memberikan ringkasan statistik deskriptif untuk setiap kolom numerik dalam dataset, seperti nilai rata-rata (mean), standar deviasi (std), nilai minimum (min), kuartil (25%, 50%, 75%), dan nilai maksimum

(max).

b) Cek Missing Value

```
[85] df.isnull().sum()
```

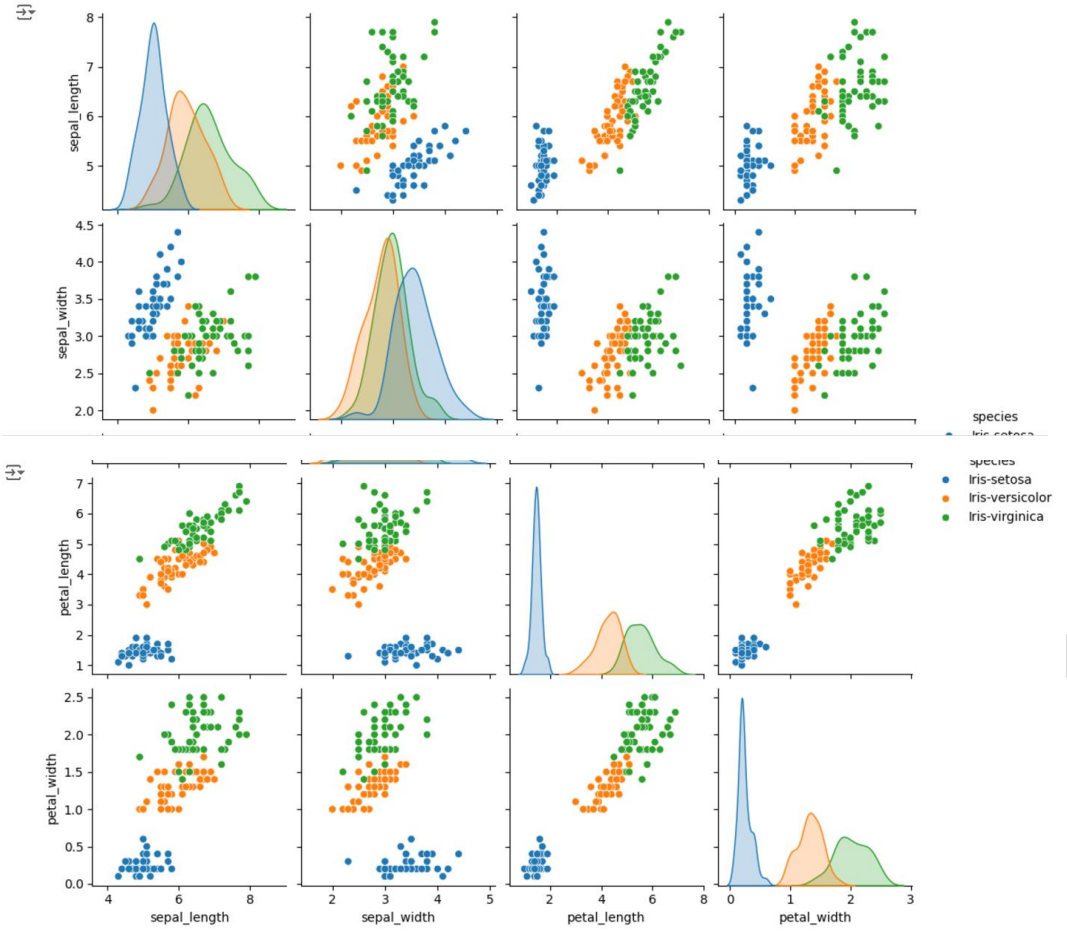
	0
sepal_length	0
sepal_width	0
petal_length	0
petal_width	0
species	0

dtype: int64

Mengidentifikasi jumlah nilai kosong (missing values) dalam setiap kolom dataset.

c) Distribusi Variabel

```
[86] sns.pairplot(df, hue="species")
     plt.show()
```



Insights: Pairplot memperlihatkan hubungan antar fitur untuk setiap spesies

Membuat scatter plot untuk semua pasangan fitur dalam dataset, dikelompokkan berdasarkan variabel species.

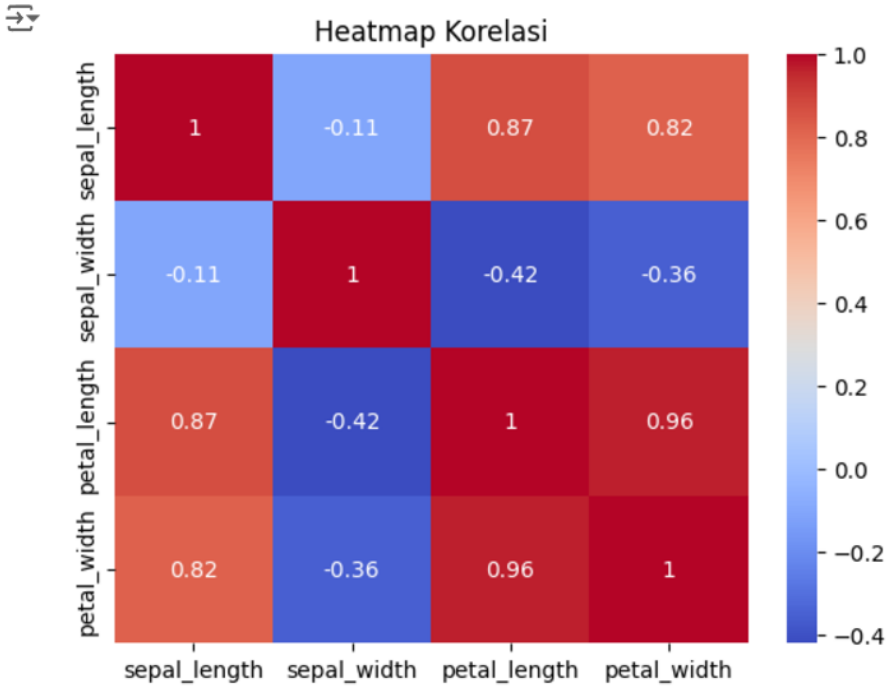
Data Visualization

a) Heatmap Korelasi

```
[87] # Menghapus kolom 'species' sebelum menghitung korelasi
     correlation = df.drop('species', axis=1).corr()
```

Menghitung matriks korelasi antar fitur numerik, dengan menghapus kolom species karena bukan numerik.

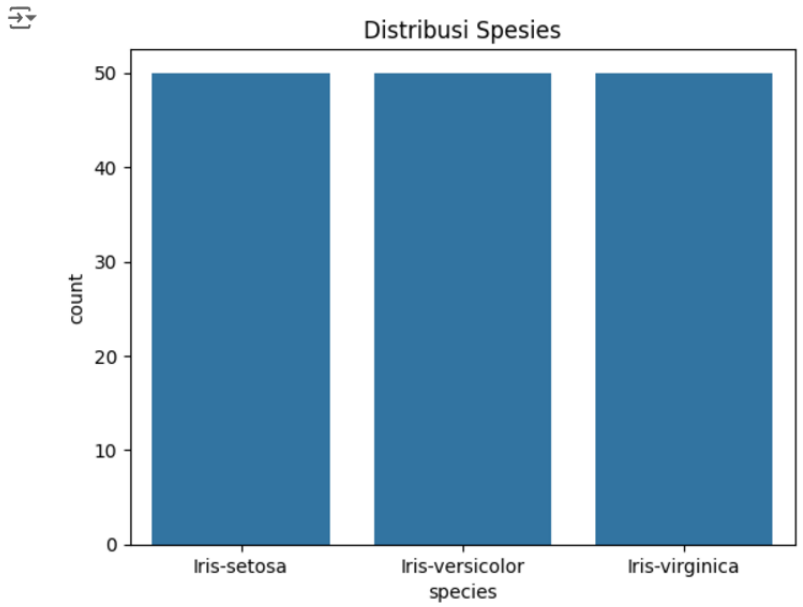
```
[88] # Visualisasi heatmap korelasi
     sns.heatmap(correlation, annot=True, cmap="coolwarm")
     plt.title("Heatmap Korelasi")
     plt.show()
```



Insights: Menunjukkan hubungan kuat antara panjang dan lebar petal.

Memvisualisasikan matriks korelasi menggunakan heatmap.

```
sns.countplot(data=df, x='species')
plt.title("Distribusi Spesies")
plt.show()
```



Insights: Setiap kelas memiliki distribusi yang seimbang.

Membuat bar chart untuk menunjukkan jumlah data di setiap kategori species.

Preprocessing

a) Pisahkan Fitur dan Target

```
[90] X = df.drop("species", axis=1)
     y = df["species"]
```

b) Split Data

```
[91] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

c) Standardisasi Fitur

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- Memisahkan dataset menjadi fitur independen (X) dan target yang akan diprediksi (y).
- Membagi data menjadi data pelatihan (80%) dan data pengujian (20%).
- Mengubah data agar memiliki rata-rata 0 dan standar deviasi 1 (standar)

Pipeline Model

a) Logistic Regression

```
[93] pipeline_lr = Pipeline([
         ('model', LogisticRegression())
     ])
```

```
[94] param_grid_lr = {
         'model__C': [0.1, 1, 10]
     }
```

```
[95] grid_lr = GridSearchCV(pipeline_lr, param_grid_lr, cv=5)
     grid_lr.fit(X_train, y_train)

     print(f"Best Params Logistic Regression: {grid_lr.best_params_}")
```

```
Best Params Logistic Regression: {'model__C': 1}
```

- Membuat pipeline untuk Logistic Regression. Pipeline membantu menyusun alur kerja machine learning secara modular.
- Mencari parameter terbaik untuk Logistic Regression menggunakan GridSearchCV.
- Menampilkan kombinasi parameter terbaik berdasarkan skor validasi.

Decision Tree

```
[96] pipeline_dt = Pipeline([
         ('model', DecisionTreeClassifier())
     ])
```

```
[97] param_grid_dt = {
         'model__max_depth': [3, 5, 10],
         'model__min_samples_split': [2, 5, 10]
     }
```

```
[98] grid_dt = GridSearchCV(pipeline_dt, param_grid_dt, cv=5)
     grid_dt.fit(X_train, y_train)

     print(f"Best Params Decision Tree: {grid_dt.best_params_}")
```

```
Best Params Decision Tree: {'model__max_depth': 3, 'model__min_samples_split': 2}
```

- Membuat pipeline untuk Decision Tree Classifier.
- Mencari parameter terbaik untuk Decision Tree.

c) k-NN

```
[99] pipeline_knn = Pipeline([
         ('model', KNeighborsClassifier())
     ])
```

```
[100] param_grid_knn = {
          'model__n_neighbors': [3, 5, 7],
          'model__weights': ['uniform', 'distance']
      }
```

```
[101] grid_knn = GridSearchCV(pipeline_knn, param_grid_knn, cv=5)
      grid_knn.fit(X_train, y_train)

      print(f"Best Params k-NN: {grid_knn.best_params_}")
```

```
Best Params k-NN: {'model__n_neighbors': 3, 'model__weights': 'uniform'}
```

- Membuat pipeline untuk k-Nearest Neighbors (k-NN).
- Mencari parameter terbaik untuk k-NN.

d) XGBoost

```
from xgboost import XGBClassifier

# Menggunakan XGBClassifier tanpa parameter use_label_encoder
xgb_model = XGBClassifier()
```

```
[103]
# Menyusun pipeline (pastikan tanpa use_label_encoder)
pipeline_xgb = Pipeline([
    ('scaler', StandardScaler()),  # Jika perlu scaling
    ('classifier', xgb_model)
])
```

```
[104] # Hyperparameter grid tanpa 'use_label_encoder'
param_grid_xgb = {
    'classifier__max_depth': [3, 5, 7],
    'classifier__learning_rate': [0.01, 0.1, 0.2],
    'classifier__n_estimators': [50, 100, 200]
}
```

```
[111] # Menerapkan GridSearchCV
grid_xgb = GridSearchCV(pipeline_xgb, param_grid_xgb, cv=5)
grid_xgb.fit(X_train, y_train_encoded)

print(f"Best Params XGBoost: {grid_xgb.best_params_}")
```

```
Best Params XGBoost: {'classifier__learning_rate': 0.01, 'classifier__max_depth': 3, 'classifier__n_estimators': 200}
```

- Membuat pipeline untuk model XGBoost.
- Mencari parameter terbaik untuk XGBoost.