1️⃣ Install & Setup Kaggle API

```
!pip install -q kagglehub[pandas-datasets]
```

```
import os

# GANTI dengan token kamu
os.environ["KAGGLE_USERNAME"] = "USERNAME"
os.environ["KAGGLE_API_KEY"] = "API_TOKEN"
```

2️⃣ Load Dataset Iris dari Kaggle

```
import kagglehub
from kagglehub import KaggleDatasetAdapter
import pandas as pd

df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,
    "uciml/iris",
    "Iris.csv"
)

df.head()
```

```
/tmp/ipython-input-496450166.py:5: DeprecationWarning: Use dataset_load() instead of load_dataset(). load_dataset() will be
  df = kagglehub.load_dataset(
Downloading from https://www.kaggle.com/api/v1/datasets/download/uciml/iris?dataset_version_number=2&file_name=Iris.csv...
100%|██████████| 4.99k/4.99k [00:00<00:00, 6.49MB/s]
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

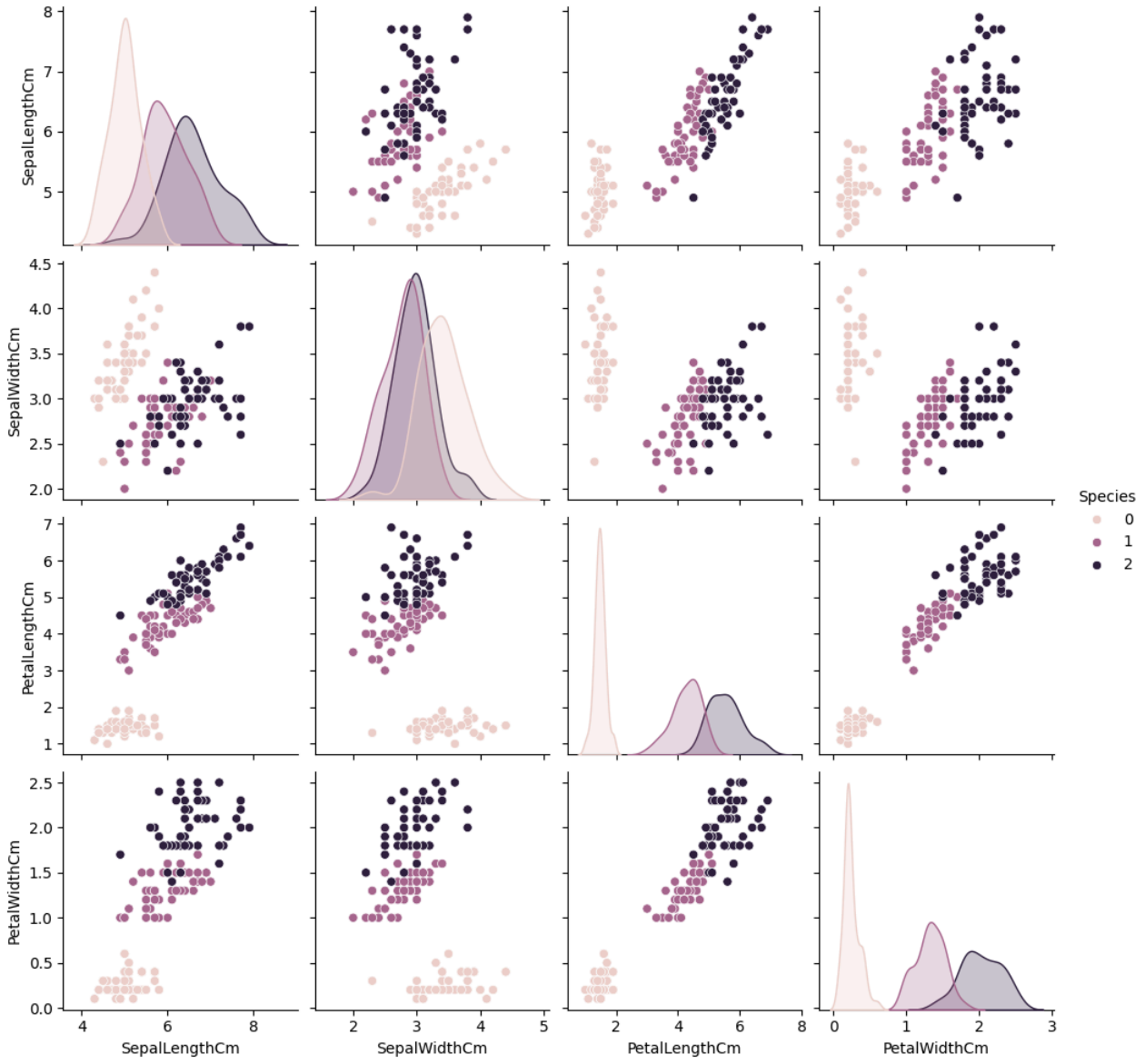
3️⃣ Data Cleaning & Preparation

```
# Drop kolom ID (tidak berguna untuk ML)
df = df.drop(columns=["Id"])

# Encode label
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df["Species"] = le.fit_transform(df["Species"])

df.head()
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

4 Exploratory Data Analysis (EDA)

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(df, hue="Species")
plt.show()
```



5 Split Data

```
from sklearn.model_selection import train_test_split

X = df.drop(columns=["Species"])
y = df["Species"]
```

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

6 Train 3 Models

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

models = {
    "KNN": KNeighborsClassifier(n_neighbors=5),
    "SVM": SVC(kernel="rbf"),
    "Logistic Regression": LogisticRegression(max_iter=200)
}

for name, model in models.items():
    model.fit(X_train, y_train)
```

7 Evaluation & Confusion Matrix

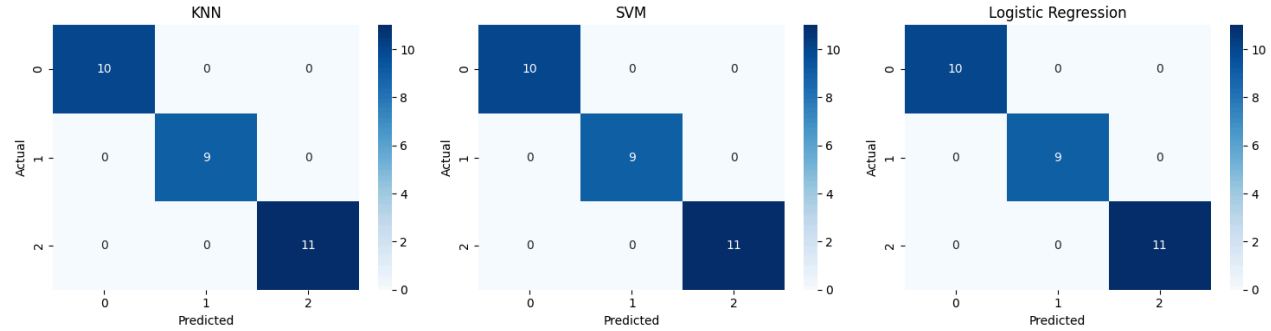
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

results = {}

for name, model in models.items():
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    results[name] = acc

    print(f"\n{name}")
    print("Accuracy:", acc)
    print(classification_report(y_test, y_pred))
```

```
KNN
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


SVM
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Logistic Regression
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

8 Visualisasi Confusion Matrix

```
plt.figure(figsize=(15,4))

for i, (name, model) in enumerate(models.items()):
    plt.subplot(1,3,i+1)
```

```
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)

    sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")
    plt.title(name)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")

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

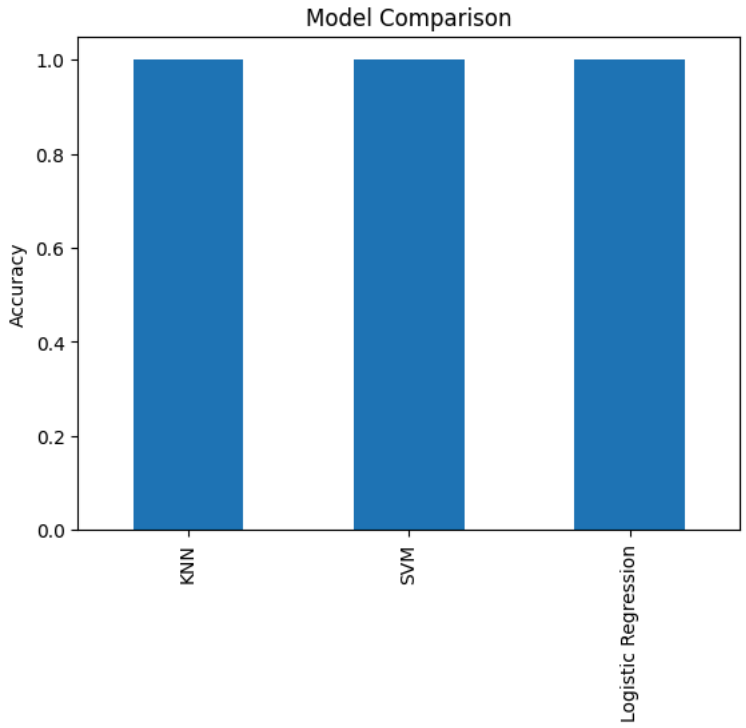


9️⃣ Perbandingan Akurasi

```
results_df = pd.DataFrame.from_dict(
    results, orient="index", columns=["Accuracy"]
)

results_df.sort_values(by="Accuracy", ascending=False)
```

	Accuracy
KNN	1.0
SVM	1.0
Logistic Regression	1.0

```
results_df.plot(kind="bar", legend=False)
plt.ylabel("Accuracy")
plt.title("Model Comparison")
plt.show()
```



The analysis of the model results shows that all three models (K-Nearest Neighbors, Support Vector Machine, and Logistic Regression)