```
import pandas as pd
import numpy as np
# Load dataset
df = pd.read_csv('/content/LabReport02-knn/IRIS.csv')
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

Next steps: Generate code with df | New interactive sheet

Confusion Matrix for Best Model

```
#Handle Missing Values

df.isnull().sum()

for col in df.columns[:-1]:  # exclude target column
    df[col].fillna(df[col].mean(), inplace=True)
#Feature & Target Separation
    X = df.drop('species', axis=1)
y = df['species']
#Min-Max Scaling
from sklearn.preprocessing import MinMaxScaler

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

Show hidden output

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    precision_score,
    recall_score,
    f1_score
)
split_ratios = [0.5, 0.4, 0.3, 0.2]   # test sizes
k_values = [1, 3, 5, 7, 9]

results = []
```

Train & Evaluate Model

```
for test_size in split_ratios:
    X_train, X_test, y_train, y_test = train_test_split(
        X_scaled, y, test_size=test_size, random_state=42
    )

    for k in k_values:
        model = KNeighborsClassifier(n_neighbors=k)
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)

        acc = accuracy_score(y_test, y_pred)
        prec = precision_score(y_test, y_pred, average='macro')
        rec = recall_score(y_test, y_pred, average='macro')
```

```
        f1 = f1_score(y_test, y_pred, average='macro')

        results.append({
            'Test Split': f"{int((1-test_size)*100)}:{int(test_size*100)}",
            'K': k,
            'Accuracy': acc,
            'Precision': prec,
            'Recall': rec,
            'F1-score': f1
        })
```

```
results_df = pd.DataFrame(results)
results_df
```

	Test Split	K	Accuracy	Precision	Recall	F1-score
0	50:50	1	0.986667	0.986111	0.985507	0.985500
1	50:50	3	0.986667	0.986111	0.985507	0.985500
2	50:50	5	0.986667	0.986111	0.985507	0.985500
3	50:50	7	0.986667	0.986111	0.985507	0.985500
4	50:50	9	0.973333	0.973333	0.971014	0.970960
5	60:40	1	0.983333	0.983333	0.981481	0.981929
6	60:40	3	0.983333	0.983333	0.981481	0.981929
7	60:40	5	0.983333	0.983333	0.981481	0.981929
8	60:40	7	0.983333	0.983333	0.981481	0.981929
9	60:40	9	0.983333	0.983333	0.981481	0.981929
10	70:30	1	1.000000	1.000000	1.000000	1.000000
11	70:30	3	1.000000	1.000000	1.000000	1.000000
12	70:30	5	1.000000	1.000000	1.000000	1.000000
13	70:30	7	1.000000	1.000000	1.000000	1.000000
14	70:30	9	1.000000	1.000000	1.000000	1.000000
15	80:20	1	1.000000	1.000000	1.000000	1.000000
16	80:20	3	1.000000	1.000000	1.000000	1.000000
17	80:20	5	1.000000	1.000000	1.000000	1.000000
18	80:20	7	1.000000	1.000000	1.000000	1.000000
19	80:20	9	1.000000	1.000000	1.000000	1.000000

Next steps: Generate code with `results_df` New interactive sheet

```
best_model = results_df.loc[results_df['Accuracy'].idxmax()]
best_model
```

	10
Test Split	70:30
K	1
Accuracy	1.0
Precision	1.0
Recall	1.0
F1-score	1.0

dtype: object

```
best_k = best_model['K']
```

```
best_split = best_model['Test Split']

test_size = int(best_split.split(':')[1]) / 100

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=test_size, random_state=42
)

best_knn = KNeighborsClassifier(n_neighbors=int(best_k))
best_knn.fit(X_train, y_train)
y_pred = best_knn.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[19,  0,  0],
       [ 0, 13,  0],
       [ 0,  0, 13]])
```