```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", for
```

```
import pandas as pd
```

```
df = pd.read_csv('/content/drive/MyDrive/AI - 2025/dataset/Iris.csv')
```

```
df.head()
```

	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

Next steps: Generate code with df New interactive sheet

```
df.columns
```

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

```
df.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
df['Species'].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
df['Species'].value_counts()
```

	count
Species	
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

dtype: int64

```
if 'Id' in df.columns:
  df = df.drop(columns=['Id'])
```

```
X = df.drop('Species', axis=1)
y = df['Species']
```

```
X.shape, y.shape
```

```
((150, 4), (150,))
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)
```

```
X_train.shape, X_test.shape
```

```
((120, 4), (30, 4))
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
import joblib
from sklearn.metrics import accuracy_score
```

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

```
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred = model.predict(X_test)
```

```
accuracy_score = accuracy_score(y_pred, y_test)
```

```
print(f"Accuarcy of Model is: {(accuracy_score * 100):.2f}%")
```

```
Accuarcy of Model is: 96.67%
```

```
joblib.dump(model, "iris_classification_model.pkl")
```

```
['iris_classification_model.pkl']
```

```
joblib.dump(scaler, "iris_classification_scaler.pkl")
```

```
['iris_classification_scaler.pkl']
```

```
Start coding or generate with AI.
```