```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

```
df = pd.read_csv('/content/Iris.csv')
if 'Id' in df.columns:
    df.drop('Id', axis=1, inplace=True)

df.head()
```

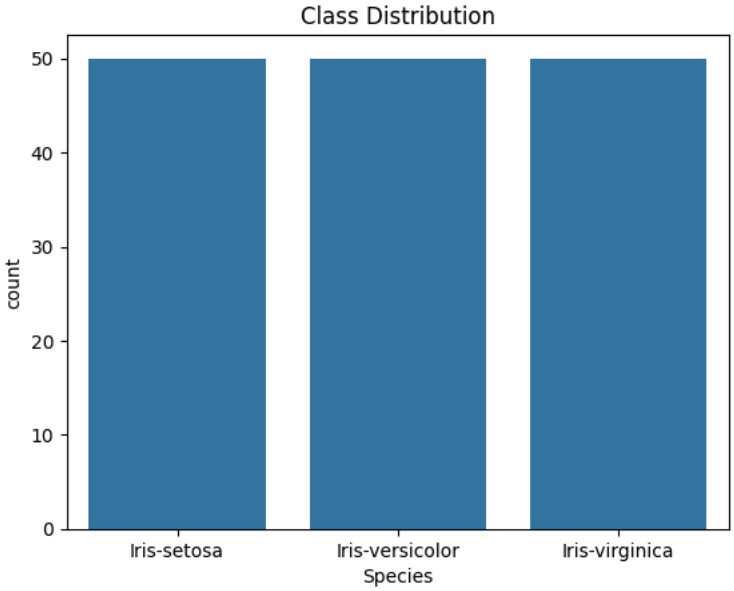
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

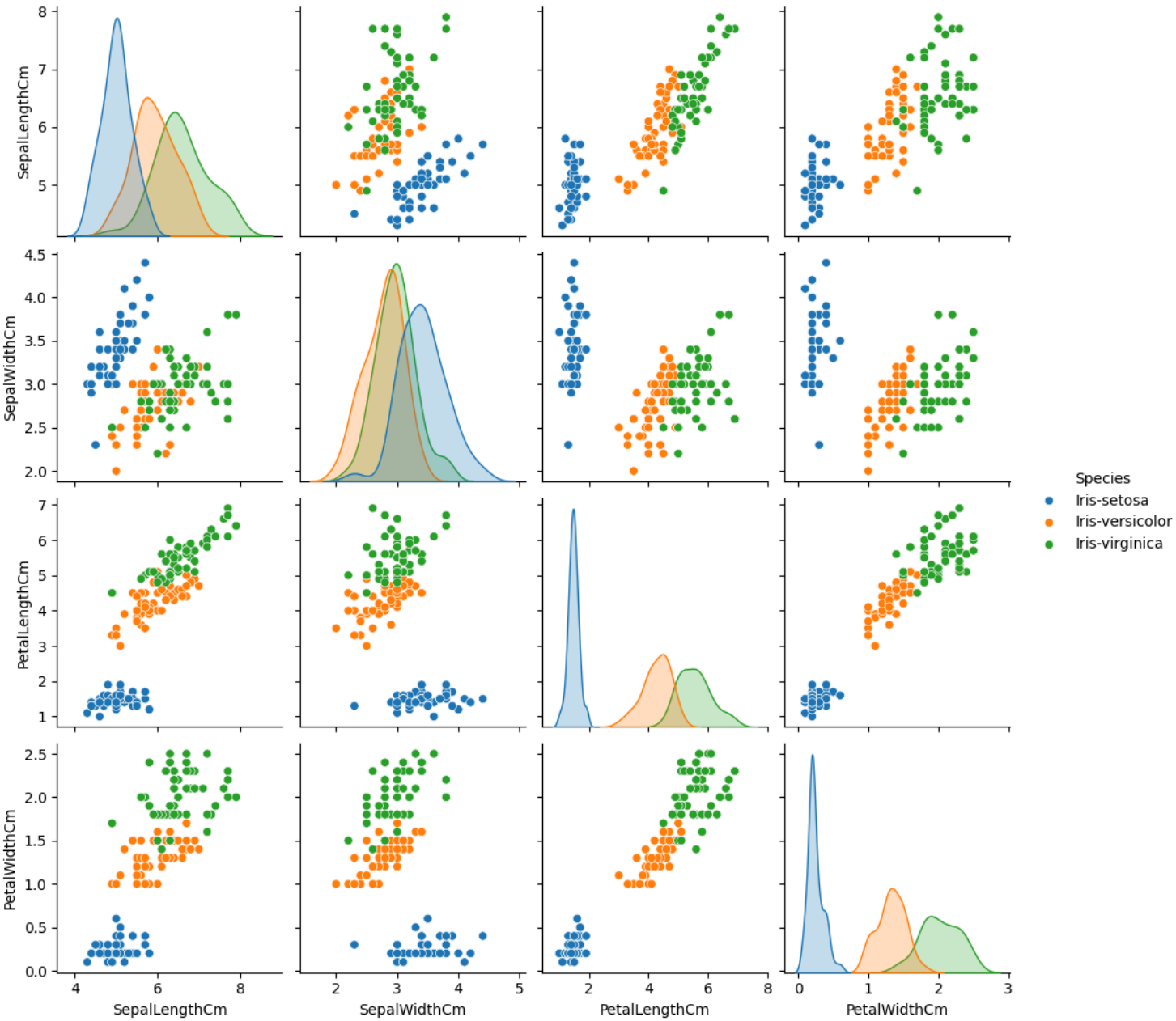
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df.info())
print("\nMissing values:\n", df.isnull().sum())
print("\nDuplicates:", df.duplicated().sum())
```

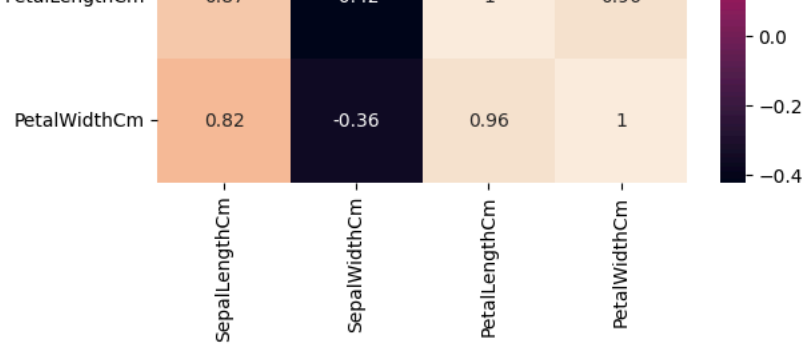
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

Missing values:
 SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64

Duplicates: 3
```

```
sns.countplot(x='Species', data=df)
plt.title("Class Distribution")
plt.show()

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

sns.heatmap(df.corr(numeric_only=True), annot=True)
plt.title("Feature Correlation")
plt.show()
```


Class Distribution
count
Species
Iris-setosa
Iris-versicolor
Iris-virginica


SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
Iris-setosa
Iris-versicolor
Iris-virginica


Feature Correlation
SepalLengthCm
SepalWidthCm
PetalLengthCm
1
-0.11
0.87
0.82
-0.11
1
-0.42
-0.36
0.87
-0.42
1
0.96




```
Q1 = df['SepalWidthCm'].quantile(0.25)
Q3 = df['SepalWidthCm'].quantile(0.75)
IQR = Q3 - Q1
df = df[(df['SepalWidthCm'] >= (Q1 - 1.5 * IQR)) & (df['SepalWidthCm'] <= (Q3 + 1.5 * IQR))]

print("After outlier removal:", df.shape)
```

```
After outlier removal: (146, 5)
```

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

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

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

```
models = {
    "Logistic Regression": LogisticRegression(),
    "KNN": KNeighborsClassifier(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name}")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
          accuracy                           0.93        30
         macro avg       0.92      0.92      0.92        30
```

```
 Iris-virginica       0.90      0.90      0.90        10

       accuracy                           0.93        30
      macro avg       0.92      0.92      0.92        30
   weighted avg       0.93      0.93      0.93        30
```

```
SVM
Accuracy: 0.9666666666666667
Confusion Matrix:
 [[12  0  0]
 [ 0  8  0]
 [ 0  1  9]]
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       0.89      1.00      0.94         8
 Iris-virginica       1.00      0.90      0.95        10

       accuracy                           0.97        30
      macro avg       0.96      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```

```
param_grid = {'n_neighbors': list(range(1, 11))}
grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid.fit(X_train, y_train)

print("Best K:", grid.best_params_)
print("Best Cross-Validation Score:", grid.best_score_)
```

```
Best K: {'n_neighbors': 7}
Best Cross-Validation Score: 0.9572463768115942
```

```
rf_scores = cross_val_score(RandomForestClassifier(), X_scaled, y, cv=5)
print("Random Forest CV Accuracy: {:.2f}".format(rf_scores.mean()))
```

```
Random Forest CV Accuracy: 0.96
```

```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

pd.Series(rf.feature_importances_, index=X.columns).plot(kind='barh')
plt.title("Feature Importance")
plt.show()
```