Tasks

- linear kernel SVC
- non-linear kernel SVC
- linear SVC with OvR (One-vs-Rest)

1.Import libraries

```
# Import necessary libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

2.Load dataset

```
# Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Labels

# Display the first 5 rows of the dataset
df = pd.DataFrame(X, columns=iris.feature_names)
df['Species'] = y
print("First 5 rows of the Iris dataset:")
print(df.head())
```

```
First 5 rows of the Iris dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   Species
0        0
1        0
2        0
3        0
4        0
```

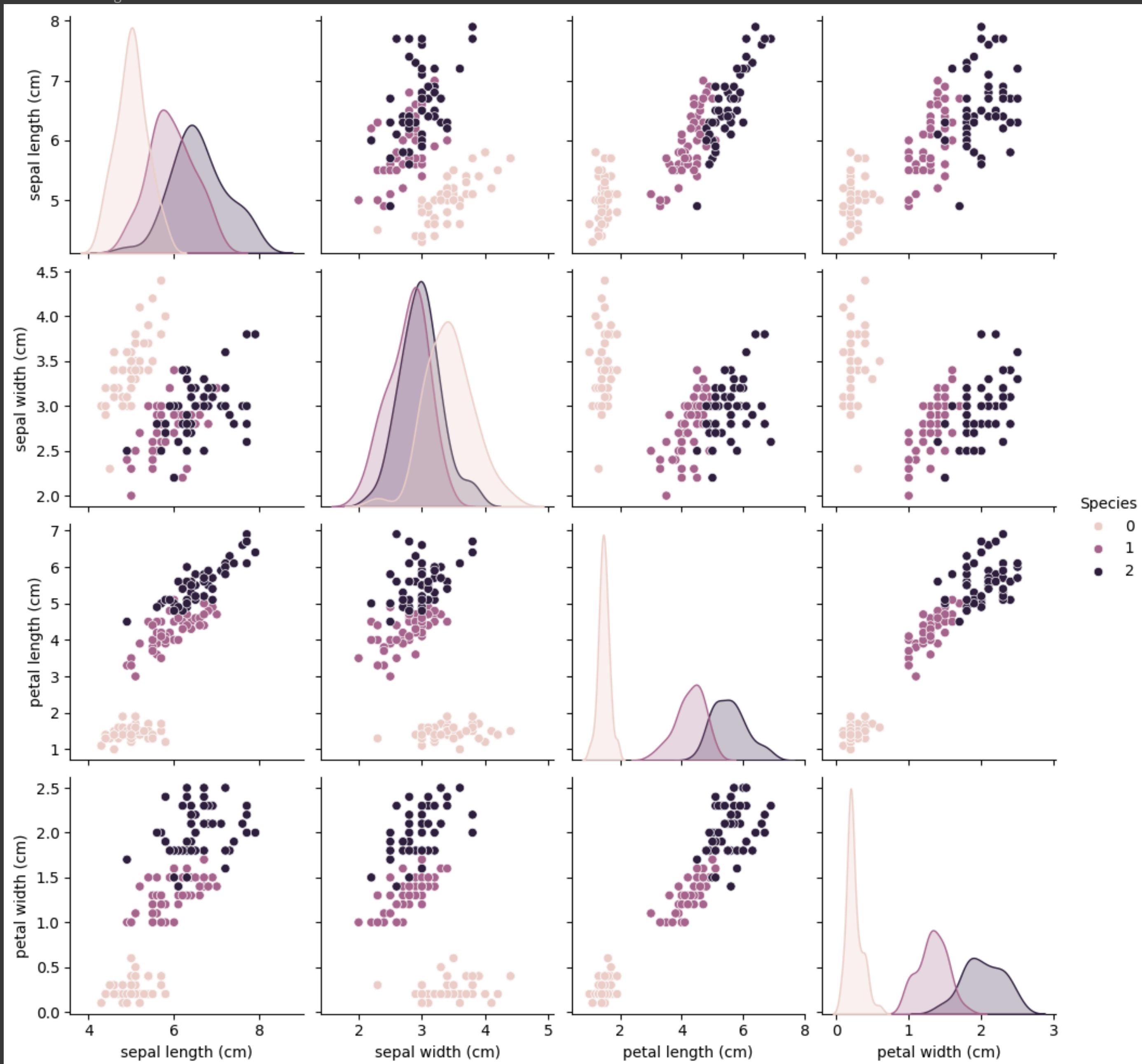
```
df['Species'].value_counts()
```

Species	count
0	50
1	50
2	50

dtype: int64

```
sns.pairplot(data = df, hue = 'Species')
```

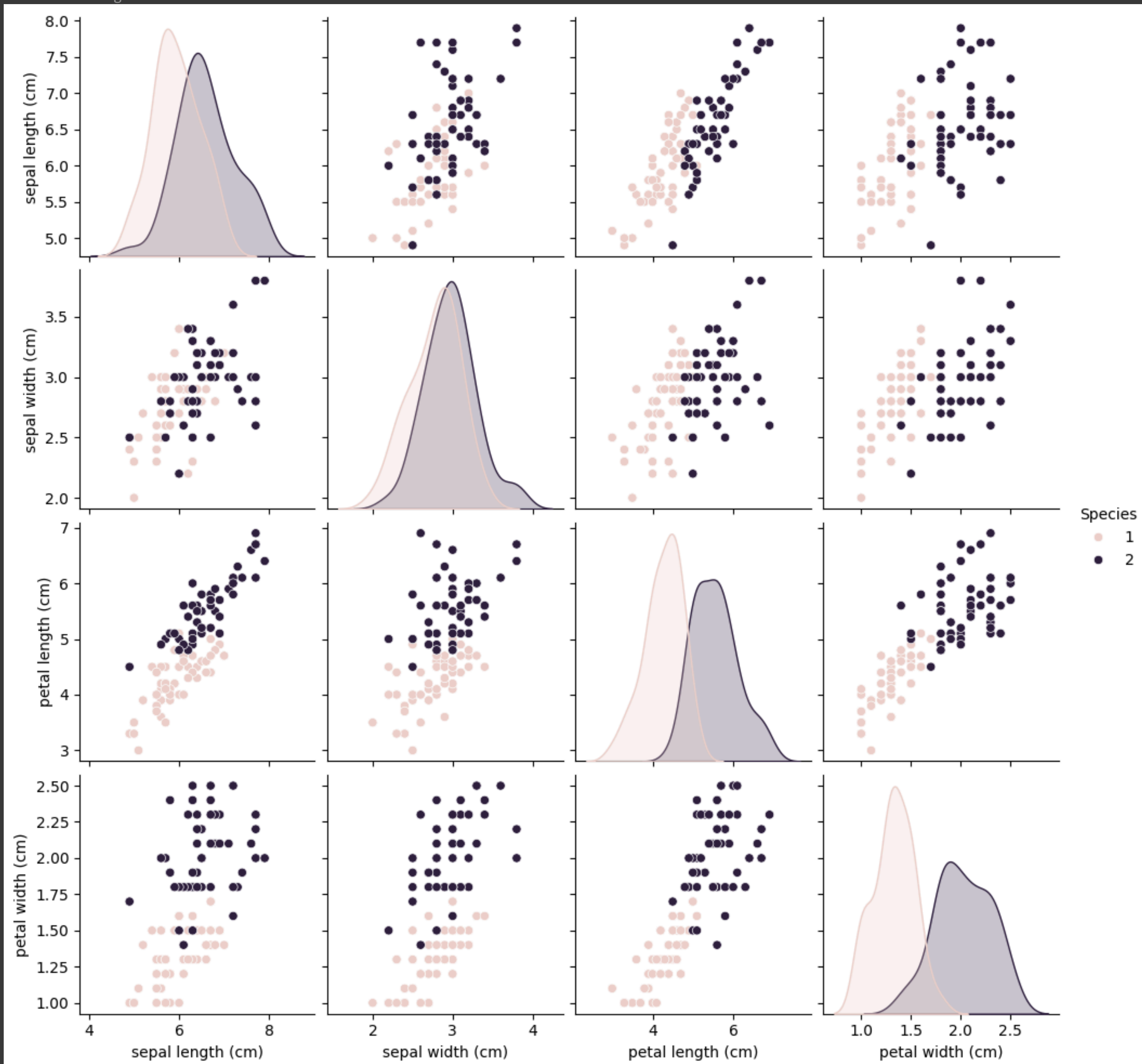
<seaborn.axisgrid.PairGrid at 0x7b3b1fab6e10>




```
# Choose 2 classes for binary classfication
df_2class = df[df['Species'] !=0]
X = df_2class.drop('Species', axis = 1)
y = df_2class['Species']
```

```
sns.pairplot(data = df_2class, hue = 'Species')
```

```
<seaborn.axisgrid.PairGrid at 0x7b3b1ae254d0>
```



3. Preprocess dataset

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize the features (important for SVM and Logistic Regression)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

4.SVM for Classification (SVC)

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

```
# Train and evaluate SVM
print("\nSupport Vector Machine (SVM):")
svm = SVC(kernel='linear', random_state=42)  # Use a linear kernel for simplicity
svm.fit(X_train, y_train)

# Predict on the test set
y_pred_svm = svm.predict(X_test)

# Evaluate accuracy
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f"Accuracy of SVM: {accuracy_svm:.2f}")

# Display SVM parameters
print("\nSVM Parameters:")
print(f"Support Vectors: {svm.support_vectors_.shape}")
print(f"Number of Support Vectors: {len(svm.support_)}")
```

```
Support Vector Machine (SVM):
Accuracy of SVM: 0.90

SVM Parameters:
Support Vectors: (9, 4)
Number of Support Vectors: 9
```

```
# Classification report

print("\nClassification Report for SVM:")
print(classification_report(y_test, y_pred_svm, target_names=iris.target_names[1:3]))
```

```
Classification Report for SVM:
              precision    recall  f1-score   support

  versicolor       1.00      0.82      0.90        17
   virginica       0.81      1.00      0.90        13

    accuracy                           0.90        30
   macro avg       0.91      0.91      0.90        30
weighted avg       0.92      0.90      0.90        30
```

5.Non-linear kernel for SVM

Use a non-linear kernel (e.g. rbf)

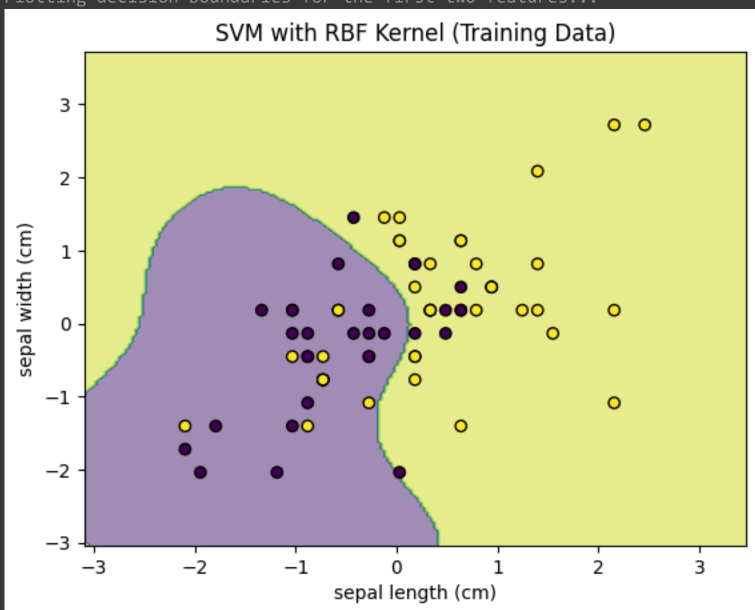
```
# Train SVM with RBF kernel
print("\nTraining SVM with RBF kernel...")
svm_rbf = SVC(kernel='rbf', gamma='scale', C=1.0, random_state=42)  # RBF kernel
svm_rbf.fit(X_train, y_train)

# Predict on the test set
y_pred = svm_rbf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy of SVM with RBF kernel: {accuracy:.2f}")

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names[1:3]))
```

```
Training SVM with RBF kernel...

Accuracy of SVM with RBF kernel: 0.87

Classification Report:
              precision    recall  f1-score   support

  versicolor       0.84      0.94      0.89        17
   virginica       0.91      0.77      0.83        13
```

```
    accuracy                           0.87        30
   macro avg       0.88      0.86      0.86        30
weighted avg       0.87      0.87      0.86        30
```

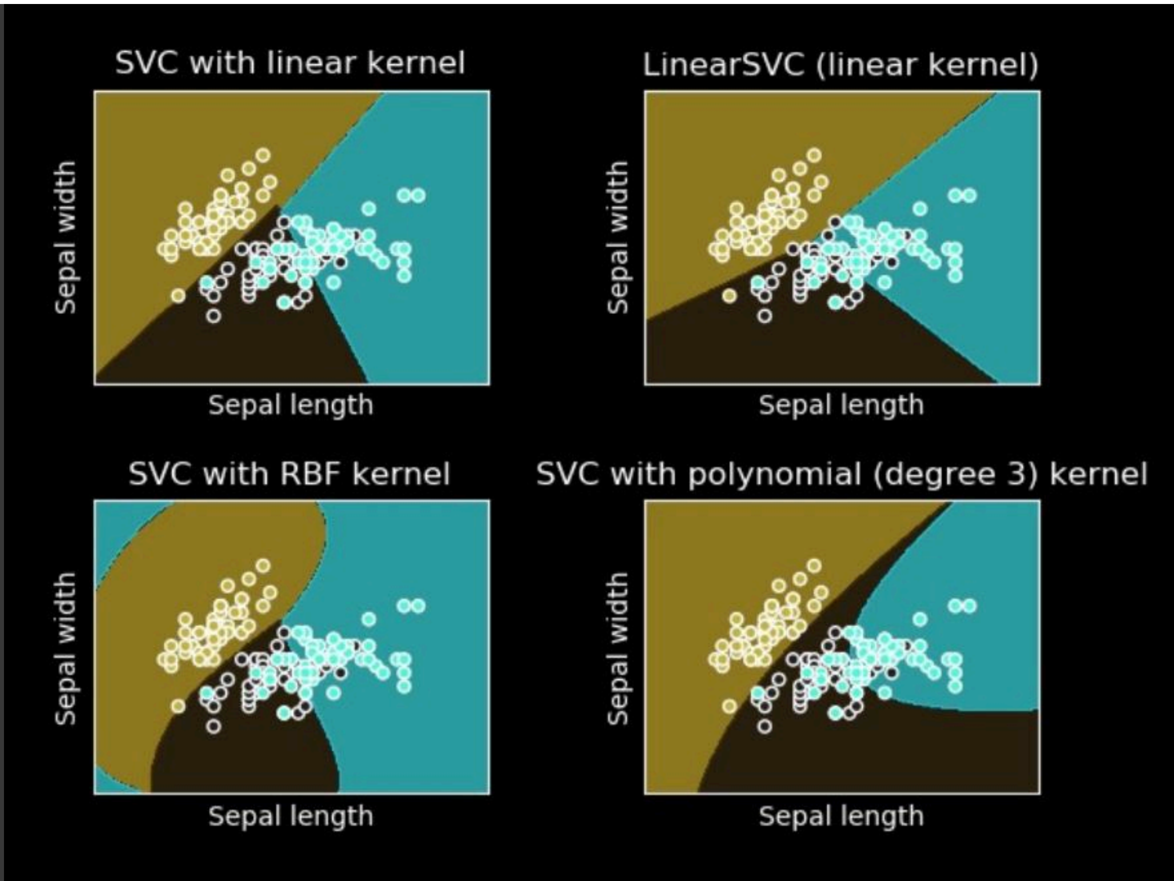
```
# Visualize decision boundaries (for 2D visualization)
from sklearn.inspection import DecisionBoundaryDisplay

def plot_decision_boundaries(X, y, model, title):
    feature_1, feature_2 = 0, 1  # Use the first two features for visualization
    X_2d = X[:, [feature_1, feature_2]]
    model.fit(X_2d, y)  # Retrain the model on the 2D subset

    disp = DecisionBoundaryDisplay.from_estimator(
        model, X_2d, response_method="predict",
        xlabel=iris.feature_names[feature_1], ylabel=iris.feature_names[feature_2],
        alpha=0.5, grid_resolution=200
    )
    disp.ax_.scatter(X_2d[:, 0], X_2d[:, 1], c=y, edgecolor="k")
    plt.title(title)
    plt.show()


# Plot decision boundaries for the first two features
print("\nPlotting decision boundaries for the first two features...")
plot_decision_boundaries(X_train, y_train, svm_rbf, "SVM with RBF Kernel (Training Data)")
```

```
Plotting decision boundaries for the first two features...
```



Your work

Use SVM algorithm with linear and non-linear kernels (poly, rbf, sigmoid) for all the 3 classes in Iris dataset.Plot the data pionts and decision boundaries for each kernel. Take the sepal length as X-axis and the sepal width as y-axis.



Complete the remaining steps and send the notebook to zwu009@citymail.cuny.edu by 5:00 pm Feb 13, 2025.

```
import matplotlib.pyplot as plt

from sklearn import datasets, svm
from sklearn.inspection import DecisionBoundaryDisplay

iris = load_iris()
X = iris.data[:, :2]
y = iris.target

models = (
    svm.SVC(kernel='linear', random_state=42),
    svm.SVC(kernel="poly", degree=3, gamma="auto", C=1.0),
    svm.SVC(kernel='rbf', gamma='scale', C=1.0, random_state=42),
    svm.SVC(kernel="sigmoid", gamma='scale', coef0=0),
)
models = (clf.fit(X, y) for clf in models)

titles = (
    "SVC with linear kernel",
    "SVC with polynomial (degree 3) kernel",
    "SVC with RBF kernel",
    "SVC with sigmoid kernel",
)

fig, sub = plt.subplots(2, 2)
plt.subplots_adjust(wspace=0.5, hspace=0.5)

X0, X1 = X[:, 0], X[:, 1]

for clf, title, ax in zip(models, titles, sub.flatten()):
    disp = DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        response_method="predict",
        cmap=plt.cm.YlGnBu,
        alpha=0.8,
        ax=ax,
        xlabel=iris.feature_names[0],
        ylabel=iris.feature_names[1],
```

```
    )
    ax.scatter(X0, X1, c=y, cmap=plt.cm.YlGnBu, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)

plt.show()
```