In [ ]:

```python
from sklearn import svm
from sklearn import metrics
from sklearn.utils import resample
from sklearn.preprocessing import scale
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.decomposition import PCA
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from matplotlib import cm
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

#import and look at dataset
iris_setosa = pd.read_csv('/Users/dcmac14/Documents/GitHub/Machine-Learning-2023/Datasets/iris-1.csv')
print(iris_setosa)

#Make a column that holds whether the flower is iris_setosa or not in binary
print(iris_setosa["Species"])
iris_setosa['Iris-setosa'] = np.where(iris_setosa['Species'] == 'Iris-setosa', 1, 0)

#make sure all the columns have proper datatypes
iris_setosa.dtypes
iris_setosa['Iris-setosa'].unique() #checking that there are only 1's and 0's in the new column
iris_setosa.drop('Species', axis=1, inplace=True) #drop old species column (no longer relevent)
iris_setosa.drop('Id', axis=1, inplace=True)

iris_setosa_petal = iris_setosa[['PetalLengthCm', 'PetalWidthCm', 'Iris-setosa']].copy()
iris_setosa_sepal = iris_setosa[['SepalLengthCm', "SepalWidthCm", 'Iris-setosa']].copy()

sns.pairplot(iris_setosa_sepal, hue='Iris-setosa', palette='bright')
sns.pairplot(iris_setosa_petal, hue='Iris-setosa', palette='bright')

print('petal dataset, ', iris_setosa_petal)
print('sepal dataset, ', iris_setosa_sepal)

#prepping for SVM by created seperate dataframes for the predictors and response variables for petal df
Xp = iris_setosa_petal.drop('Iris-setosa', axis=1).copy() #predictor variable set without iris setosa column
Xp.head
yp = iris_setosa_petal['Iris-setosa'].copy() #response variable just iris setosa column
yp.head

#prepping for SVM by created seperate dataframes for the predictors and response variables for sepal df
Xs = iris_setosa_sepal.drop('Iris-setosa', axis=1).copy() #predictor variable set without iris setosa column
Xs.head
ys = iris_setosa_sepal['Iris-setosa'].copy() #response variable just iris setosa column
ys.head

#splitting the data into 30% test and 70% train, then scaling data and fitting to model for petal df
X_trainp, X_testp, y_trainp, y_testp = train_test_split(Xp,yp, test_size=0.30)
X_trainp_scaled = scale(X_trainp)
X_testp_scaled = scale(X_testp)

clfp_svm = SVC(kernel='linear') #classifier support vector machine with linear kernel
clfp_svm.fit(X_trainp_scaled, y_trainp) #fitting the scaled x train, y train data to the SVM

print(clfp_svm.support_vectors_)


#splitting the data into 30% test and 70% train, then scaling data and fitting to model for sepal df
X_trains, X_tests, y_trains, y_tests = train_test_split(Xs,ys, test_size=0.30)
X_trains_scaled = scale(X_trains)
X_tests_scaled = scale(X_tests)

clfs_svm = SVC(kernel='linear') #classifier support vector machine with linear kernel
clfs_svm.fit(X_trains_scaled, y_trains) #fitting the scaled x train, y train data to the SVM

print(clfs_svm.support_vectors_)


#confusion matrix - create y predicted values then make confusion matrix for the test data result. Petal DF
y_predp = clfp_svm.predict(X_testp_scaled)
cmp = confusion_matrix(y_testp, y_predp)
print(cmp)
disp = ConfusionMatrixDisplay(confusion_matrix=cmp,display_labels=["Not Iris Setosa - PetalDF", 'Iris Setosa - Pet
disp.plot()

#confusion matrix - create y predicted values then make confusion matrix for the test data result. Sepal DF
y_preds = clfs_svm.predict(X_tests_scaled)
cms = confusion_matrix(y_tests, y_preds)
print(cms)
disp = ConfusionMatrixDisplay(confusion_matrix=cms,display_labels=["Not Iris Setosa - SepalDF", 'Iris Setosa - Sep
disp.plot()


plt.figure(figsize=(10, 8))
# Plotting our two-features-space - petaldf
sns.scatterplot(x=X_trainp_scaled[:, 0],
                y=X_trainp_scaled[:, 1],
                hue=y_trainp,
                s=8);
# Constructing a hyperplane using a formula.
wp = clfp_svm.coef_[0]           # w consists of 2 elements
bp = clfp_svm.intercept_[0]      # b consists of 1 element
x_points_p = np.linspace(-2, 2)    # generating x-points from -1 to 1
y_points_p = -(wp[0] / wp[1]) * x_points_p - bp / wp[1]  # getting corresponding y-points
# Plotting a red hyperplane
plt.plot(x_points_p, y_points_p, c='r');

# plot results - petaldf

# Encircle support vectors
plt.scatter(clfp_svm.support_vectors_[:, 0],
            clfp_svm.support_vectors_[:, 1],
            s=50,
            facecolors='none',
            edgecolors='k',
            alpha=.5);
# Step 2 (unit-vector):
w_hat_p = clfp_svm.coef_[0] / (np.sqrt(np.sum(clfp_svm.coef_[0] ** 2)))
# Step 3 (margin):
margin_p = 1 / np.sqrt(np.sum(clfp_svm.coef_[0] ** 2))
# Step 4 (calculate points of the margin lines):
decision_boundary_points_p = np.array(list(zip(x_points_p, y_points_p)))
points_of_line_above_p = decision_boundary_points_p + w_hat_p * margin_p
points_of_line_below_p = decision_boundary_points_p - w_hat_p * margin_p
# Plot margin lines
# Blue margin line above
plt.plot(points_of_line_above_p[:, 0],
         points_of_line_above_p[:, 1],
         'b--',
         linewidth=2)
# Green margin line below
plt.plot(points_of_line_below_p[:, 0],
         points_of_line_below_p[:, 1],
         'g--',
         linewidth=2)

plt.figure(figsize=(10, 8))

# Plotting our two-features-space - sepaldf
sns.scatterplot(x=X_trains_scaled[:, 0],
                y=X_trains_scaled[:, 1],
                hue=y_trains,
                s=8);
# Constructing a hyperplane using a formula.
ws = clfs_svm.coef_[0]           # w consists of 2 elements
bs = clfs_svm.intercept_[0]      # b consists of 1 element
x_points_s = np.linspace(-2, 2)    # generating x-points from -1 to 1
y_points_s = -(ws[0] / ws[1]) * x_points_s - bs / ws[1]  # getting corresponding y-points
# Plotting a red hyperplane
plt.plot(x_points_s, y_points_s, c='r');


# plot results - sepaldf

# Encircle support vectors
plt.scatter(clfs_svm.support_vectors_[:, 0],
            clfs_svm.support_vectors_[:, 1],
            s=50,
            facecolors='none',
            edgecolors='k',
            alpha=.5);
# Step 2 (unit-vector):
w_hat_s = clfs_svm.coef_[0] / (np.sqrt(np.sum(clfs_svm.coef_[0] ** 2)))
# Step 3 (margin):
margin_s = 1 / np.sqrt(np.sum(clfs_svm.coef_[0] ** 2))
# Step 4 (calculate points of the margin lines):
decision_boundary_points_s = np.array(list(zip(x_points_s, y_points_s)))
points_of_line_above_s = decision_boundary_points_s + w_hat_s * margin_s
points_of_line_below_s = decision_boundary_points_s - w_hat_s * margin_s
# Plot margin lines
# Blue margin line above
plt.plot(points_of_line_above_s[:, 0],
         points_of_line_above_s[:, 1],
         'b--',
         linewidth=2)
# Green margin line below
plt.plot(points_of_line_below_s[:, 0],
         points_of_line_below_s[:, 1],
         'g--',
         linewidth=2)
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

             Species
0        Iris-setosa
1        Iris-setosa
2        Iris-setosa
3        Iris-setosa
4        Iris-setosa
..               ...
145  Not-Iris-setosa
146  Not-Iris-setosa
147  Not-Iris-setosa
148  Not-Iris-setosa
149  Not-Iris-setosa

[150 rows x 6 columns]
0          Iris-setosa
1          Iris-setosa
2          Iris-setosa
3          Iris-setosa
4          Iris-setosa
            ...
145    Not-Iris-setosa
146    Not-Iris-setosa
147    Not-Iris-setosa
148    Not-Iris-setosa
149    Not-Iris-setosa
Name: Species, Length: 150, dtype: object
petal dataset,       PetalLengthCm  PetalWidthCm  Iris-setosa
0              1.4           0.2            1
1              1.4           0.2            1
2              1.3           0.2            1
3              1.5           0.2            1
4              1.4           0.2            1
..             ...           ...          ...
145            5.2           2.3            0
146            5.0           1.9            0
147            5.2           2.0            0
148            5.4           2.3            0
149            5.1           1.8            0

[150 rows x 3 columns]
sepal dataset,       SepalLengthCm  SepalWidthCm  Iris-setosa
0              5.1           3.5            1
1              4.9           3.0            1
2              4.7           3.2            1
3              4.6           3.1            1
4              5.0           3.6            1
..             ...           ...          ...
145            6.7           3.0            0
146            6.3           2.5            0
147            6.5           3.0            0
148            6.2           3.4            0
149            5.9           3.0            0

[150 rows x 3 columns]
[[-0.25802744 -0.26158402]
 [-0.14466811 -0.26158402]
 [-1.16490214 -0.91554406]
 [-1.0515428  -1.04633606]]
[[-1.10624249 -1.17444044]
 [-0.53121545 -0.07340253]
 [-0.30120464 -0.07340253]
 [-0.30120464 -0.07340253]
 [-0.76122627 -0.73402527]
 [-1.56626412 -1.6148556 ]
 [-0.53121545  0.8074278 ]
 [-1.2212479  -0.07340253]
 [-0.41621005  1.02763538]
 [-0.99123708 -0.07340253]
 [-1.2212479  -0.07340253]]
[[30  0]
 [ 0 15]]
[[29  0]
 [ 0 16]]
```

Out[ ]: [<matplotlib.lines.Line2D at 0x161768460>]