SVM Classification

18BCS063

Sujan guptha B

1.Use apple_and_oranges.csv dataset.

Import the dataset and divide the dataset in to Training and Testing dataset.

· Apply SVM classifier (use given kernel type) to build model using Training Dataset.

Kernel Type = Radial basis function (RBF), gamma=0.8

Kernel Type = Linear

· Predict class label for data items in test Dataset.

· Print the Confusion Matrix andCalculate the accuracy of the predictions.

· Visualize the classifier results

In [80]:
```
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
df=pd.read_csv("apples_and_oranges.csv")
```

In [81]:
```
df.head()
```

Out[81]:

	Weight	Size	Class
0	69	4.39	orange
1	69	4.21	orange
2	65	4.09	orange
3	72	5.85	apple
4	67	4.70	orange

In [82]:
```
df.info
```

Out[82]:
```
<bound method DataFrame.info of     Weight  Size   Class
0       69  4.39  orange
1       69  4.21  orange
2       65  4.09  orange
3       72  5.85   apple
4       67  4.70  orange
5       73  5.68   apple
6       70  5.56   apple
7       75  5.11   apple
8       74  5.36   apple
9       65  4.27  orange
```

In [83]:
```
from sklearn.model_selection import train_test_split
```

In [84]:
```
training_set, test_set = train_test_split(df, test_size = 0.2, random_state = 1)
```

In [61]:
```
X_train = training_set.iloc[:,0:2].values
Y_train = training_set.iloc[:,2].values
X_test = test_set.iloc[:,0:2].values
Y_test = test_set.iloc[:,2].values
```

In [62]:
```
classifier = SVC(kernel='rbf', random_state = 1, gamma=0.6)
classifier.fit(X_train,Y_train)
```

Out[62]:
```
SVC(gamma=0.6, random_state=1)
```

In [63]:
```
Y_prediction = classifier.predict(X_test)
```

In [64]:
```
test_set["Predictions"] = Y_prediction
```

```
<ipython-input-64-85ee06a09b3d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_set["Predictions"] = Y_prediction
```

In [65]:
```
test_set
```

In [65]:
```
test_set
```

Out[65]:

	Weight	Size	Class	Predictions
2	65	4.09	orange	orange
31	66	4.68	orange	orange
3	72	5.85	apple	apple
21	70	4.83	orange	apple
27	70	4.22	orange	orange
29	71	5.26	apple	apple
22	69	4.61	orange	orange
39	73	5.03	apple	apple

In [66]:
```
cm = confusion_matrix(Y_test,Y_prediction)
accuracy = float(cm.diagonal().sum())/len(Y_test)
print("Accuracy Of SVM For The Given Dataset : ", accuracy)
```

```
Accuracy Of SVM For The Given Dataset :  0.875
```

In [67]:
```
le = LabelEncoder()
Y_train = le.fit_transform(Y_train)
```

In [68]:
```
classifier = SVC(kernel='rbf', random_state = 1, gamma=0.6)
classifier.fit(X_train,Y_train)
```

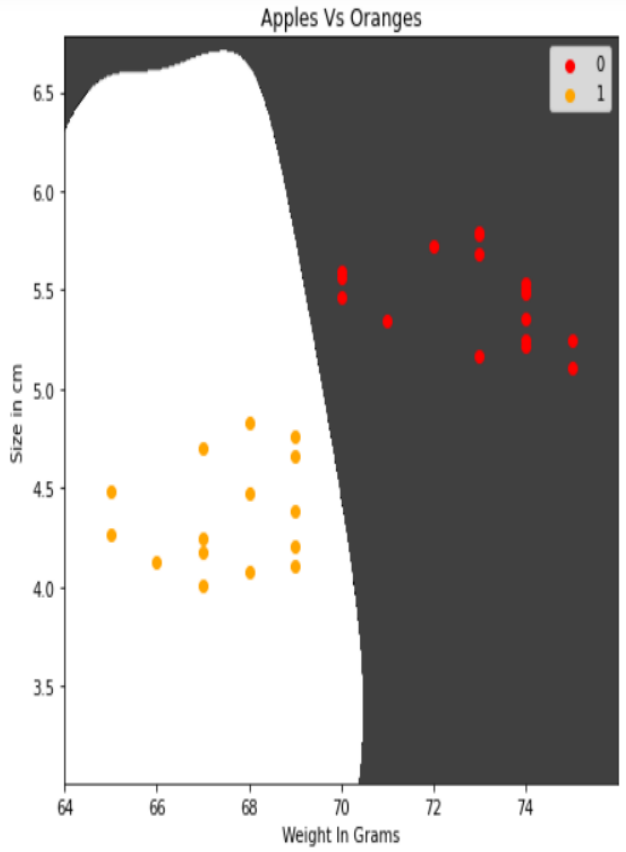
Out[68]:
```
SVC(gamma=0.6, random_state=1)
```

In [70]:
```
#Actual Output
```

In [71]:
```
plt.figure(figsize = (7,7))
X_set, y_set = X_train, Y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01), np.arange(start = X_set
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColor
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'orange'))(i), label = j)
plt.title('Apples Vs Oranges')
plt.xlabel('Weight In Grams')
plt.ylabel('Size in cm')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```

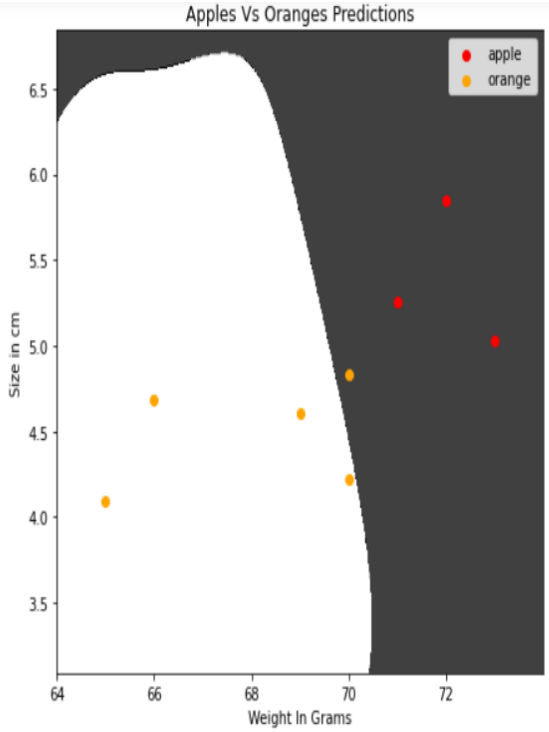
Apples Vs Oranges



In [72]:
```
# Predicted Output
```

In [73]:
```
plt.figure(figsize = (7,7))
X_set, y_set = X_test, Y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),np.arange(start = X_set[
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),alpha = 0.75, cmap = ListedColorm
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],c = ListedColormap(('red', 'orange'))(i), label = j)
plt.title('Apples Vs Oranges Predictions')
plt.xlabel('Weight In Grams')
plt.ylabel('Size in cm')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```



[85]:
```
classifier2= SVC(kernel='linear', random_state = 1)
classifier2.fit(X_train,Y_train)
```

[85]:
```
SVC(kernel='linear', random_state=1)
```

[86]:
```
Y_pred = classifier.predict(X_test)
```

In [86]:
```
Y_pred = classifier.predict(X_test)
```

In [87]:
```
test_set["Predictions"] = Y_pred
```

```
<ipython-input-87-946a65001e17>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_set["Predictions"] = Y_pred
```

In [79]:
```
test_set
```

Out[79]:

	Weight	Size	Class	Predictions
2	65	4.09	orange	1
31	66	4.68	orange	1
3	72	5.85	apple	0
21	70	4.83	orange	0
27	70	4.22	orange	1
29	71	5.26	apple	0
22	69	4.61	orange	1
39	73	5.03	apple	0

2. Use “iris.csv” dataset.

I. Import and Visualize the dataset.

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm

df=pd.read_csv("iris.csv")
```

In [8]:
```
df.head()
```

Out[8]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

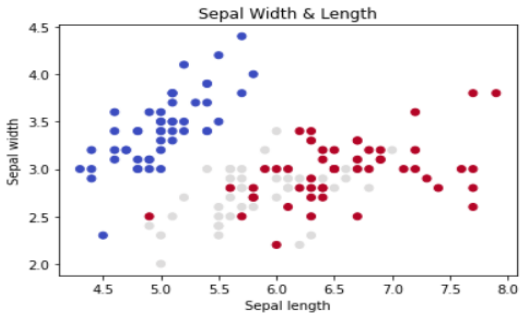
In [9]:
```
df
```

Out[9]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
...	...	...	...	...	...

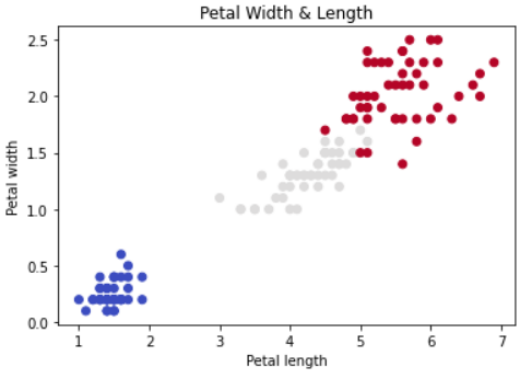
1. Visualizing the relationship between sepal width and target classes

```
In [10]: def visuvalize_sepal_data():
             iris = datasets.load_iris()
             X = iris.data[:, :2]
             y = iris.target
             plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
             plt.xlabel('Sepal length')
             plt.ylabel('Sepal width')
             plt.title('Sepal Width & Length')
             plt.show()

         visuvalize_sepal_data()
```



2. Visualizing the relationship between petal width and target classes

```
In [11]: def visuvalize_petal_data():
             iris = datasets.load_iris()
             X = iris.data[:, 2:]
             y = iris.target
             plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
             plt.xlabel('Petal length')
             plt.ylabel('Petal width')
             plt.title('Petal Width & Length')
             plt.show()

         visuvalize_petal_data()
```



II. Apply SVM (use given kernel type) for prediction.

1. Model given Kernel SVM classifier using Iris Sepal features and Visualizing the modeled SVM classifiers with Iris Sepal features

   Kernel Type = Radial basis function (RBF), gamma=0.8

   Kernel Type = Linear

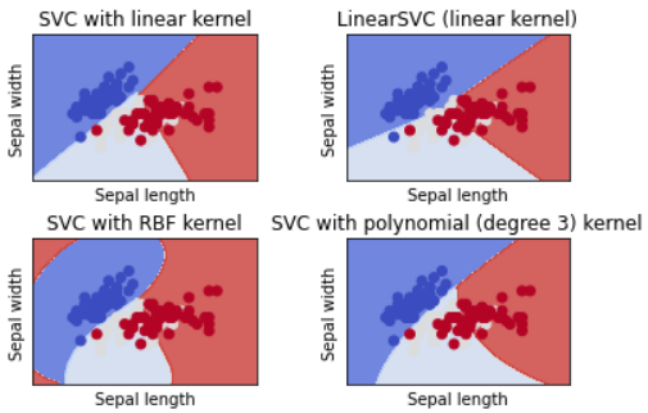
   Kernel Type = Polynomial, Degree = 3

In [12]:
```
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
C = 1.0
svc = svm.SVC(kernel='linear', C=C).fit(X, y)
lin_svc = svm.LinearSVC(C=C).fit(X, y)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
```

```
C:\Users\A B Charan\anaconda3\lib\site-packages\sklearn\svm\_base.py:976: ConvergenceWarning: Liblinear failed to converge, inc
rease the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

In [13]:
```
h = .02
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

titles = ['SVC with linear kernel',
          'LinearSVC (linear kernel)',
          'SVC with RBF kernel',
          'SVC with polynomial (degree 3) kernel']


for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):

    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
```

```
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])

plt.show()
```



2. Model given Kernel SVM classifier using Iris Sepal features and Visualizing the modeled SVM classifiers with Iris petal features

In [14]:
```
iris = datasets.load_iris()
X = iris.data[:, 2:]  # we only take the last two features.
y = iris.target
C = 1.0  # SVM regularization parameter

# SVC with linear kernel
svc = svm.SVC(kernel='linear', C=C).fit(X, y)
# LinearSVC (linear kernel)
lin_svc = svm.LinearSVC(C=C).fit(X, y)
# SVC with RBF kernel
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
# SVC with polynomial (degree 3) kernel
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
```

In [15]:
```
h = .02  # step size in the mesh
# create a mesh to plot in
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
# title for the plots
titles = ['SVC with linear kernel',
          'LinearSVC (linear kernel)',
          'SVC with RBF kernel',
          'SVC with polynomial (degree 3) kernel']


for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max]x[y_min, y_max].
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    # Plot also the training points
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    plt.xlabel('Petal length')
    plt.ylabel('Petal width')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])

plt.show()
```