

**Aim: Program to implement text classification using Support vector machine**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("apples_and_oranges.csv")
print(data.head(15))
plt.show()
```

	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
10	73	5.79	apple
11	70	5.47	apple
12	74	5.53	apple
13	68	4.47	orange
14	74	5.22	apple

In [6]:

*#Splitting the dataset into training and test samples*

```
from sklearn.model_selection import train_test_split
training_set, test_set = train_test_split(data, test_size = 0.2, random_state = 1)
```

In [7]:

*#Classifying the predictors and target*

```
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 [8]:

*#Initializing Support Vector Machine and fitting the training data*

```
from sklearn.svm import SVC
classifier = SVC(kernel='rbf', random_state =
```

```
1)classifier.fit(X_train,Y_train)
```

Out[8]:

```
SVC(random_state=1)
```

In [9]:

```
#Predicting the classes for test set
```

```
Y_pred = classifier.predict(X_test)
```

In [13]:

```
#Attaching the predictions to test set for comparing
```

```
test_set["Predictions"] = Y_predplt.show()
```

In [14]:

```
#Calculating the accuracy of the predictions
```

```
from sklearn.metrics import confusion_matrixcm =
```

```
confusion_matrix(Y_test,Y_pred)accuracy =
```

```
float(cm.diagonal().sum())/len(Y_test)print("\nAccuracy Of SVM For The  
Given Dataset : ", accuracy)
```

Accuracy Of SVM For The Given Dataset : 0.375

In [15]:

```
#Visualizing the classifier
```

```
from sklearn.preprocessing
```

```
import LabelEncoderle = LabelEncoder()Y_train = le.fit_transform(Y_train)
```

In [16]:

```
from sklearn.svm import SVCclassifier = SVC(kernel='rbf', random_state =
```

```
1)classifier.fit(X_train,Y_train)
```

Out[16]:

```
SVC(random_state=1)
```

In [21]:

```
import numpy as np
```

```
import matplotlib.pyplot as plt from matplotlib.colors
```

```
import ListedColormapplt.figure(figsize = (7,7))X_set, y_set = X_train,
```

```
Y_trainX1, 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[:, 1].min() - 1, stop =
```

```
X_set[:, 1].max() + 1, step = 0.01))plt.contourf(X1, X2,
```

```
classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha
```

```
= 0.75, cmap = ListedColormap(('black', 'white'))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 ==
```

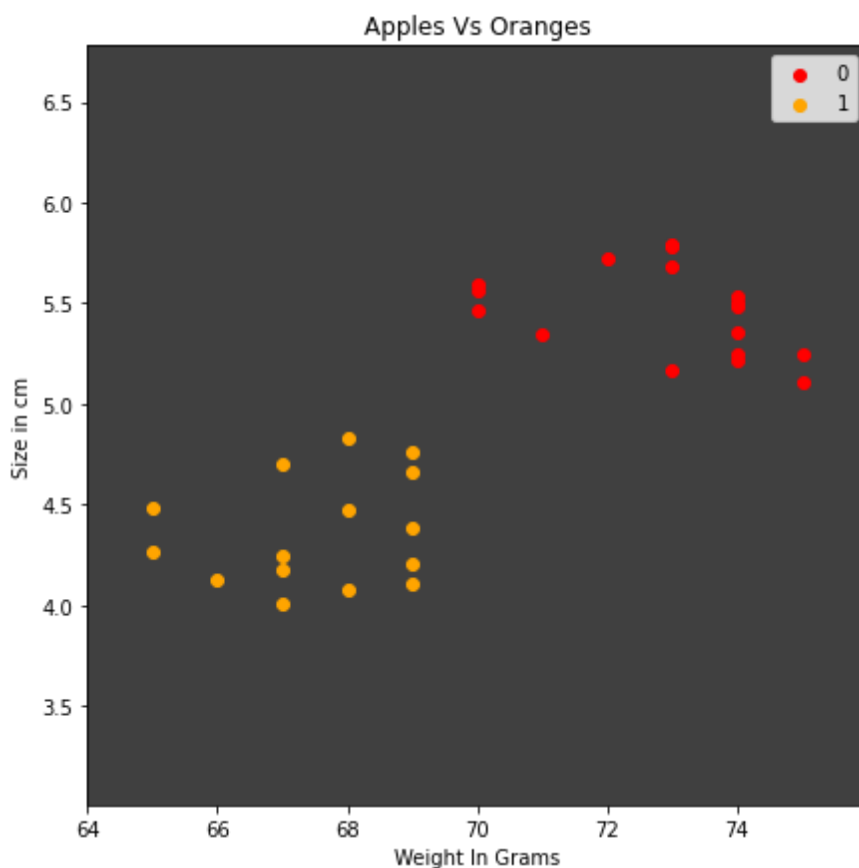
```
j, 1],c = ListedColormap(('red', 'orange'))(i), label = j)plt.title('Apples Vs
```

```
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 2D array with a single row if you 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 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.

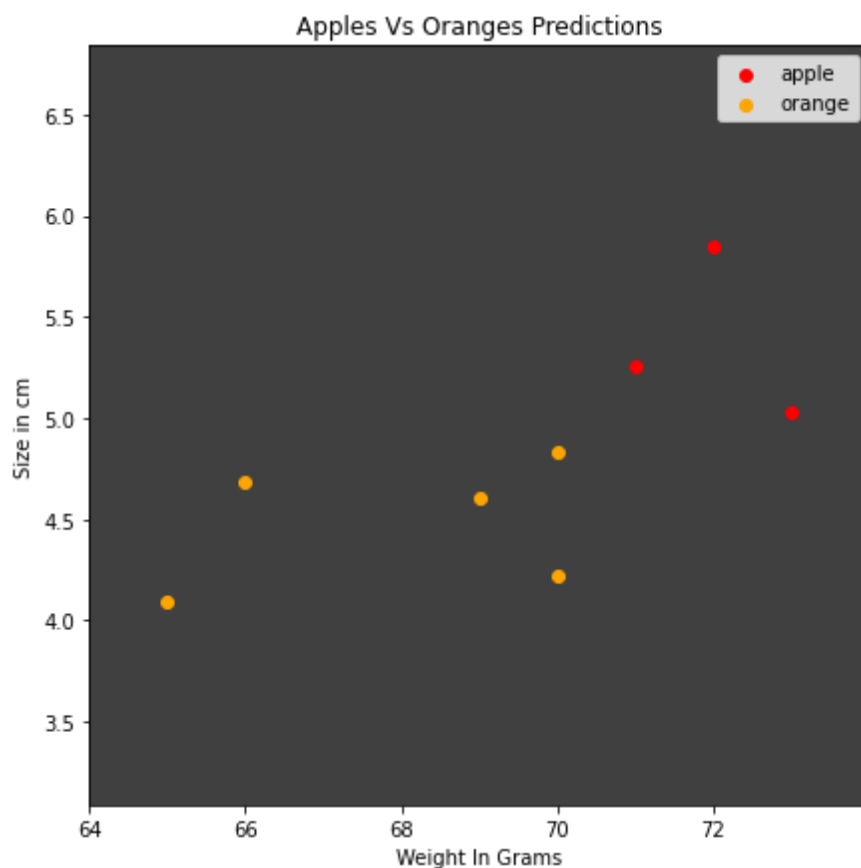


In [20]:

```
import numpy as np
import matplotlib.pyplot as plt from matplotlib.colors
import ListedColormapplt.figure(figsize = (7,7))X_set, y_set = X_test,
Y_testX1, 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[:, 1].min() - 1, stop =
X_set[:, 1].max() + 1, step = 0.01))plt.contourf(X1, X2,
classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),alpha
= 0.75, cmap = ListedColormap(('black', 'white'))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 2D array with a single row if you 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 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.



In []: