

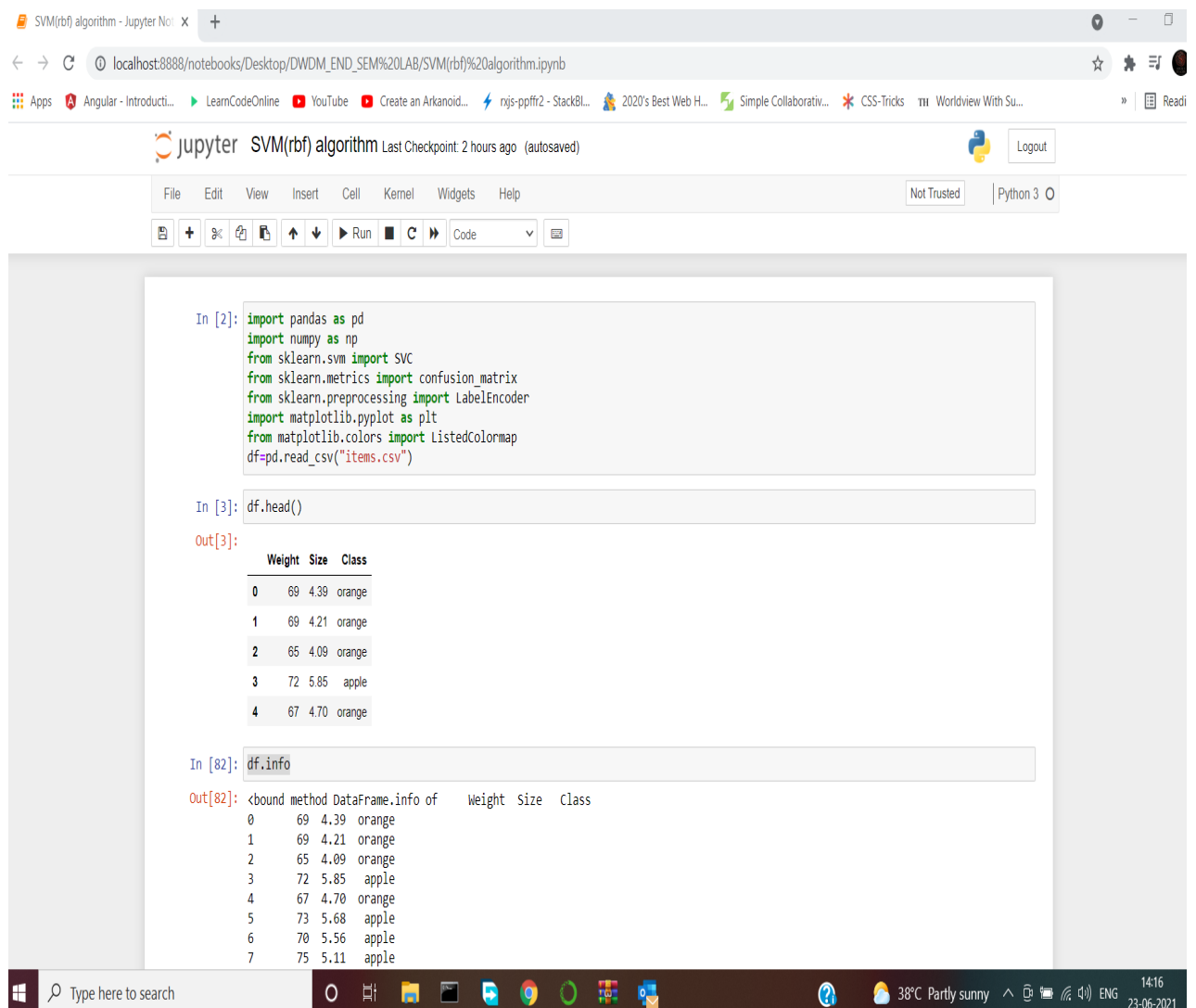
# set (3) DWDM END LAB -SVM(rbf) algorithm

## 18BCS037

## GANESH SETHU

1.items.csv dataset choosen

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



The screenshot displays a Jupyter Notebook titled "SVM(rbf) algorithm" running in a web browser at localhost:8888. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The code is written in Python 3.

```
In [2]: 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("items.csv")
```

```
In [3]: df.head()
```

Out[3]:

	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
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
```

The bottom of the image shows a Windows taskbar with various application icons, a search bar, and system status information including temperature (38°C), weather (Partly sunny), and time (14:16, 23-06-2021).

```
|: from sklearn.model_selection import train_test_split
```

```
|: training_set, test_set = train_test_split(df, test_size = 0.2, random_state = 1)
```

```
|: 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
```

```
|: classifier = SVC(kernel='rbf', random_state = 1, gamma=0.6)  
classifier.fit(X_train,Y_train)
```

```
|: SVC(gamma=0.6, random_state=1)
```

```
|: Y_prediction = classifier.predict(X_test)
```

In [11]: test\_set

Out[11]:

	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 [12]: 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 [13]: le = LabelEncoder()  
Y_train = le.fit_transform(Y_train)
```

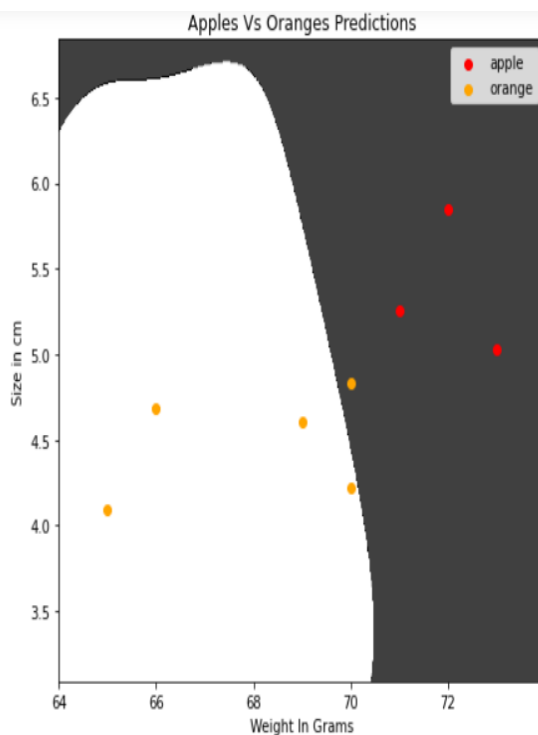
```
In [23]: classifier = SVC(kernel='rbf', random_state = 1, gamma=0.8)  
classifier.fit(X_train,Y_train)
```

Out[23]: SVC(gamma=0.8, random\_state=1)

```

In [25]: 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[:, 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(2))
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()

```



### # Predicted Output

```
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[:, 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(['red', 'orange']))
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()
```

```
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](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

