Ex no: 6

A PYTHON PROGRAM TO IMPLEMENT SVM CLASSIFIER MODEL

Aim:

To implement a SVM classifier model using python and determine its accuracy.

Algorithm:

Step 1: Import Necessary Libraries

1. Import numpy as np.

2. Import pandas as pd.

3. Import SVM from sklearn.

4. Import matplotlib.pyplot as plt.

5. Import seaborn as sns.

6. Set the font_scale attribute to 1.2 in seaborn.

Step 2: Load and Display Dataset

1. Read the dataset using `pd.read_csv()`.

2. Display the first five instances using the `head()` function.

Step 3: Plot Initial Data

1. Use the `sns.lmplot()` function.

2. Set the x and y axes to "Sugar" and "Flour".

3. Assign "recipes" to the data parameter.

4. Assign "Type" to the hue parameter.

5. Set the palette to "Set1".

6. Set fit_reg to False.

7. Set scatter_kws to {"s": 70}.

8. Plot the graph.

Step 4: Prepare Data for SVM

1. Extract "Sugar" and "Butter" columns from the recipes dataset and

assign to variable `sugar_butter`.

2. Create a new variable `type_label`.

3. For each value in the "Type" column, assign 0 if it is "Muffin" and 1 otherwise.

Step 5: Train SVM Model

1. Import the SVC module from the svm library.

2. Create an SVC model with kernel type set to linear.

3. Fit the model using `sugar_butter` and `type_label` as the parameters.

Step 6: Calculate Decision Boundary

1. Use the `model.coef_` function to get the coefficients of the linear model.

2. Assign the coefficients to a list named `w`.

3. Calculate the slope `a` as `w[0] / w[1]`.

4. Use `np.linspace()` to generate values from 5 to 30 and assign to variable `xx`.

5. Calculate the intercept using the first value of the model intercept and divide by

`w[1]`.

6. Calculate the decision boundary line `y` as `a * xx - (model.intercept_[0] / w[1])`.

Step 7: Calculate Support Vector Boundaries

1. Assign the first support vector to variable `b`.

2. Calculate `yy_down` as `a * xx + (b[1] - a * b[0])`.

3. Assign the last support vector to variable `b`.

4. Calculate `yy_up` using the same method.

Step 8: Plot Decision Boundary

1. Use the `sns.lmplot()` function again with the same parameters as in Step 3.

2. Plot the decision boundary line `xx` and `yy`.

Step 9: Plot Support Vector Boundaries

1. Plot the decision boundary with `xx`, `yy_down`, and `'k--'`.

2. Plot the support vector boundaries with `xx`, `yy_up`, and `'k--'`.

3. Scatter plot the first and last support vectors.

Step 10: Import Additional Libraries

1. Import `confusion_matrix` from `sklearn.metrics`.

2. Import `classification_report` from `sklearn.metrics`.

3. Import `train_test_split` from `sklearn.model_selection`.

Step 11: Split Dataset

1. Assign `x_train`, `x_test`, `y_train`, and `y_test` using `train_test_split`.

2. Set the test size to 0.2.

Step 12: Train New Model

1. Create a new SVC model named `model1`.

2. Fit the model using the training data (`x_train` and `y_train`).

Step 13: Make Predictions

1. Use the `predict()` function on `model1` with `x_test` as the parameter.

2. Assign the predictions to variable `pred`.

Step 14: Evaluate Model

1. Display the confusion matrix.

2. Display the classification report.

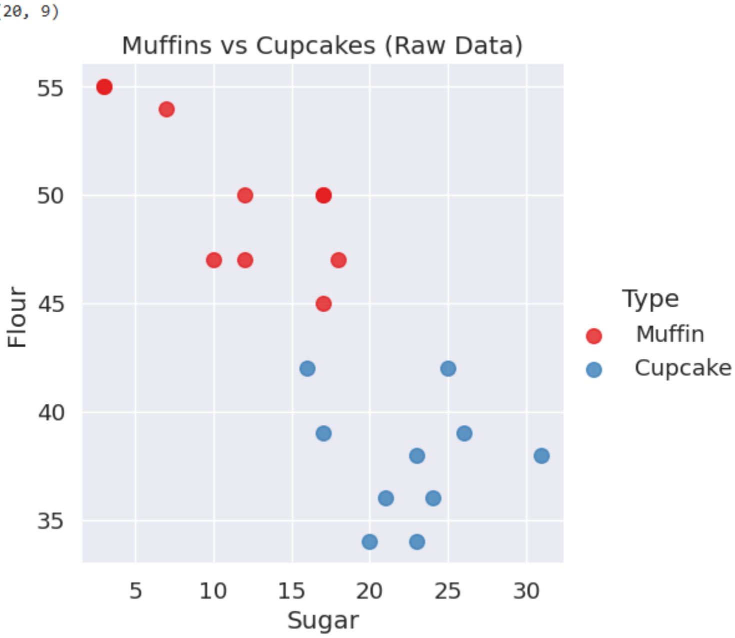
PROGRAM:

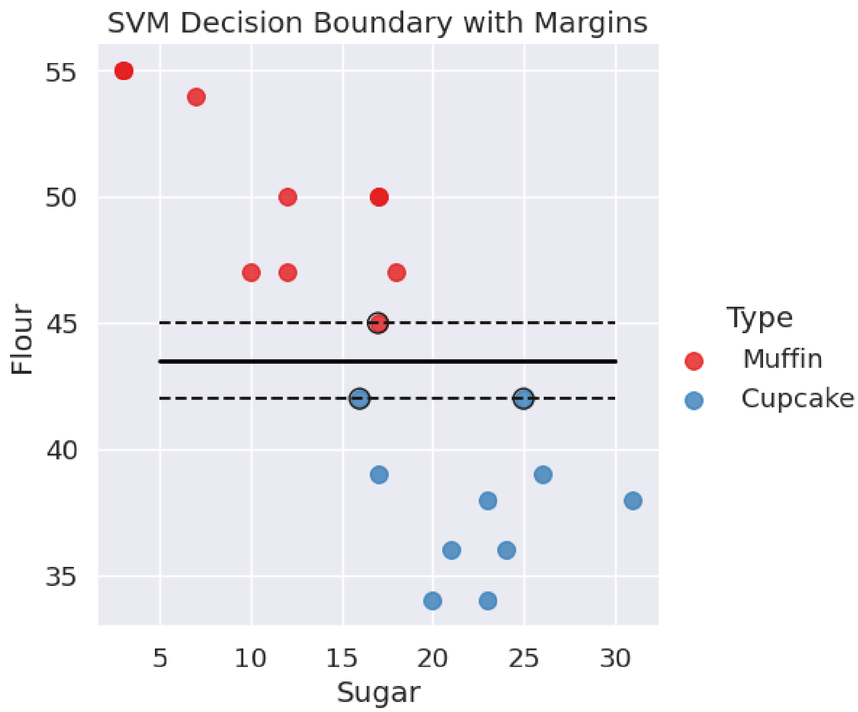
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

sns.set(font_scale=1.2)

recipes = pd.read_csv('recipes_muffins_cupcakes -
recipes_muffins_cupcakes.csv')

print(recipes.head())
print(recipes.shape)

sns.lmplot(x='Sugar', y='Flour', data=recipes, hue='Type', palette='Set1',
fit_reg=False, scatter_kws={"s": 70})
plt.title("Muffins vs Cupcakes (Raw Data)")
plt.show()

sugar_flour = recipes[['Sugar', 'Flour']].values
type_label = np.where(recipes['Type'] == 'Muffin', 0, 1)

model = svm.SVC(kernel='linear')
model.fit(sugar_flour, type_label)

w = model.coef_[0]
a = -w[0] / w[1]
xx = np.linspace(5, 30)
yy = a * xx - (model.intercept_[0] / w[1])

b = model.support_vectors_[0]
yy_down = a * xx + (b[1] - a * b[0])
b = model.support_vectors_[-1]
yy_up = a * xx + (b[1] - a * b[0])

sns.lmplot(x='Sugar', y='Flour', data=recipes, hue='Type', palette='Set1',
fit_reg=False, scatter_kws={"s": 70})
plt.plot(xx, yy, linewidth=2, color='black')
plt.plot(xx, yy_down, 'k--')
plt.plot(xx, yy_up, 'k--')
```

```
plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1],
s=100, facecolors='none', edgecolors='k')
plt.title("SVM Decision Boundary with Margins")
plt.show()

x_train, x_test, y_train, y_test = train_test_split(sugar_flour,
type_label, test_size=0.2, random_state=42)

model1 = svm.SVC(kernel='linear')
model1.fit(x_train, y_train)
pred = model1.predict(x_test)

print(pred)
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

OUTPUT :

```
     Type  Flour  Milk  Sugar  Butter  Egg  Baking Powder  Vanilla  Salt
0  Muffin     55    28      3       7    5              2        0     0
1  Muffin     47    24     12       6    9              1        0     0
2  Muffin     47    23     18       6    4              1        0     0
3  Muffin     45    11     17      17    8              1        0     0
4  Muffin     50    25     12       6    5              2        1     0
(20, 9)
```






```
[0 1 0 0]
[[2 0]
 [1 1]]
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         2
           1       1.00      0.50      0.67         2

    accuracy                           0.75         4
   macro avg       0.83      0.75      0.73         4
weighted avg       0.83      0.75      0.73         4
```

RESULT:

Thus, the Python program to implement the SVM classifier model has been executed successfully, and the classified output has been analyzed for the given dataset