

# A python program to Implement SVM classifier Model.

Aim:

To implement a SVM classifier model in 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 sns.

Step 2: Load and Display Dataset.

1. Read the dataset (muffins.csv) using 'pd.read\_csv()'

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

Step 3: Plot initial Data:

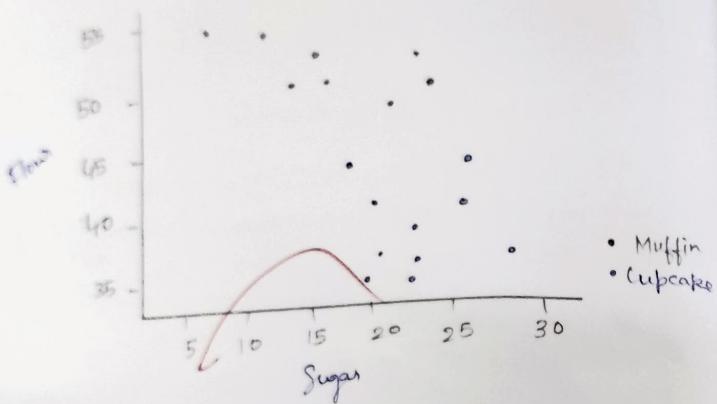
1. Use the 'sns.heatmap()' function.

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 "Set 1".



6. Set fit\_size to 50.

7. Set scatter\_size to 5 "S": 10<sup>3</sup>.

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 1 if it is "truffin" and 0 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\_coeff-' function to get the coefficients of the linear model.
2. Assign the coefficients to a list named 'w'.
3. Calculate the Slope ' $a = w[0]/w[1]$ '.
4. Use 'np.linspace()' to generate values from 5 to 30 and assign to variable 'xx'.
5. Calculate the decision boundary line 'Y' as ' $a * xx - \text{model intercept}[0] / w[1]$ '.

### Step 7: Calculate Support vector Boundaries:

1. Assign the first support vector to variable 'b'.
2. Calculate ' $yy\_down$ ' as ' $a * xx + (b[i] - a * b[0])$ '.
3. Calculate ' $yy\_up$ ' using the same method.

### Step 8: plot Decision Boundary :

1. Use the 'SVM-impl(0)' 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 and "l<--".

2. Scatter plot the first and last support vectors.

Step 10: Import additional libraries:

1. Import 'ConfusionMatrix' from

'sklearn.metrics'.

2. Import 'train-test-split' from 'sklearn-  
selection'.

Step 11: split Dataset:

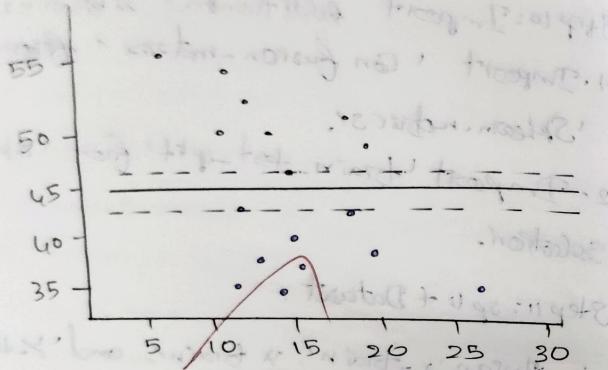
1. Assign 'x-train', 'x-train', and 'y-test'  
using 'train-test-split'.

2. Set the test size to 0.2.

Step 12: Evaluate Model:

1. Display the Confusion matrix.

2. Display the Classification report.



Program:

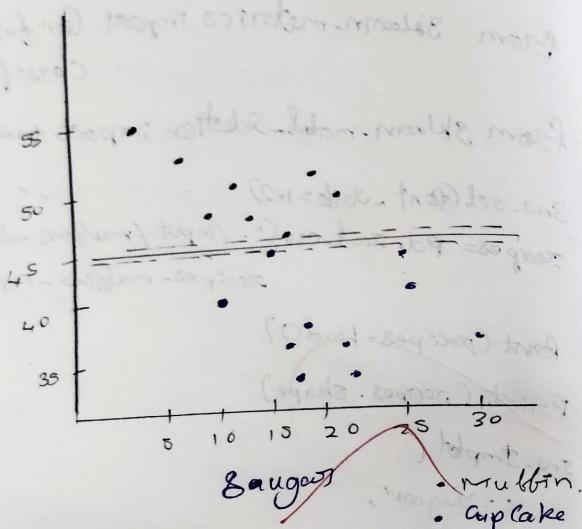
```

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

sns.set(font_scale=1.2)
recipes = pd.read_csv("../input/muffins-dataset/
                      recipes-muffins-cupcakes.csv")
print(recipes.head())
print(recipes.shape)
sns.set()
x = 'Sugar',
y = 'Flour',
data = recipes,
hue = 'Type',
palette = 'Set1',
fit_kreg = False,
Scatter_kws = {'s': 70}

```

)



`sugar_flour = recipes[['Sugar', 'Flour']].values`  
`type_label = np.where(recipes['Type'] == 'Muffin', 0)`  
`w = model.coef_[0]`  
`a = -w[0] / w[1]`  
`xx = np.linspace(5, 30)`  
`yy = a * xx - (model.intercept_[0] / w[1])`  
`b_down = model.support_vectors_[0]`  
`yy_down = a * xx + (b_down[1] - a * b_down[0])`

### Sns. Implot [

`x = 'Sugar'`,

`y = 'Flour'`,

`data = recipes`

`hue = 'Type'`,

`palette = 'Set1'`,

`fit_reg = False,`

`scatter_kws = {"s": 70}`

)

plt. scatter(

model. Support\_vectors\_[:, 0],

model. Support\_vectors\_[:, 1],

s=80,

facecolor='none',

edgecolor='k',

)

plt.show()

x-train, x-test, y-train, y-test = train-test split

. shuffle, 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("Predictions:", pred)

print("In Confusion Matrix:\n", confusion-

matrix(x-test, pred))

print("In Classification Report:\n", classification-

report(x-test, pred))

Result:

Thus the Python program to implement SVM classifier model has been executed successfully and the class label output has been analyzed for the given dataset (mushroom.csv).