

# 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-size attribute to 12 in seaborn.

Step 2: Load and Display Dataset.

1. Read the dataset (muffins.csv) using 'pd.read\_csv()'.
2. Display the first five rows using the 'head()' function.

Step 3: Plot Initial Data:

1. Use the 'sns.plt()' function.
2. Set the X and Y axes to "Sugar" and "Flow".
3. Assign "sugar" to the data parameter.
4. Assign "Type" to the hue parameter.
5. Set the palette to "Set 2".

6. Set fit - msg to False.

7. Set scatter - kms 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 if it is "butter" 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 6 to 30 and assign to variable 'xx'.
5. 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. Calculate 'yy-up' using the same method.

### Step 8: Plot Decision Boundary:

1. Use the 'sns.scatter()' 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 "kc--".
2. Scatter plot the first and last support vectors.

Step 10: Import additional libraries:

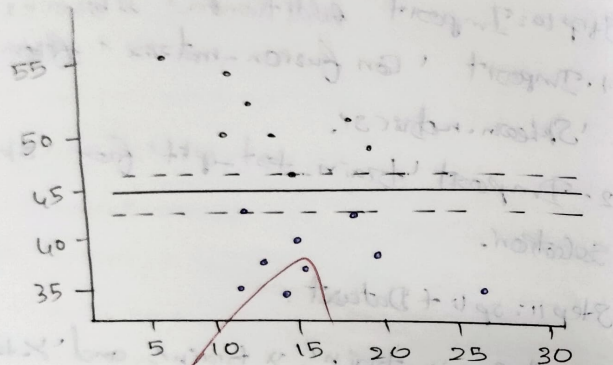
1. Import 'Confusion-matrix' from 'sklearn.metrics'.
2. Import 'train-test-split' from 'sklearn-selection'.

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 ConfusionMatrix,
classfication_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.plot
```

```
x = 'Sugar',
```

```
y = 'Flour',
```

```
data = recipes,
```

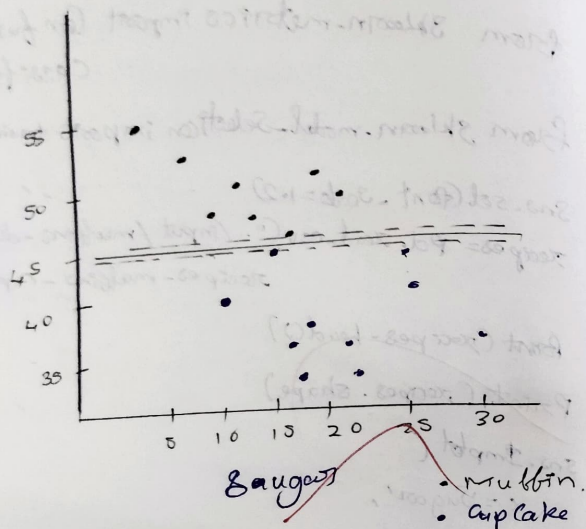
```
hue = 'Type',
```

```
palette = 'set1',
```

```
fit_reg = False,
```

```
scatter_kws = {'s': 70}
```

✓



```
Sugar-Flow = rscapes(['Sugar', 'Flow']).values
type-label = np.where(rscapes['type'] == 'mutton', 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 = 'Flow',
```

```
data = rscapes
```

```
hue = 'Type',
```

```
palette = 'Set 1',
```

```
fit_reg = False,
```

```
scatter_kws = dict(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(  
    sugarflow, 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("\n Confusion Matrix:\n", confusion-  
        matrix(x_test, pred))
```

```
print("\n Classification Report:\n", classification-  
        report(x_test, pred))
```

For

To implement a decision tree using a python program for the given dataset and get the trained decision tree.

Algorithm:

Step 1: Import the Data Dataset

- Import 'load\_breast\_glass' sklearn dataset.

Step 2: Import necessary libraries

- Import numpy as np.

- Import matplotlib.pyplot as plt.

Step 3: Declare and initialize parameters

- Declare and initialize n, class no.

- Declare and initialize 'features' and 'target'.

Result:

Thus the python program to implement svm classifier model has been executed successfully and the classified output has been analyzed for the given dataset (multiclass).