A PYTHON PROGRAM TO IMPLEMENT SVM CLASSIFIER MODEL

Ex.No.: 6

AIM:-

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

ALGORITHM:-

Step1: Import all the necessary libraries(numpy as np, pandas as pd, svm from sklearn, matplotlib.pyplot as plt and seaborn as sns and set the “font_scale” attribute to “1.2”.

Step2: Read the dataset(muffins.csv) using the “pd.read_csv()” function and display the first five instances using the “head()” function.

Step3: Using the “sns.lmplot()” function and plot the graph, where the inputs are “Sugar” and “Flour”. Assign “recipes” to “data”, “Type” to “hue”, “Set1” to “palette”,”False” to “fit_reg” and “{“s”:70}” to “sctter_kws”.

Step4: Assign the values of “Sugar” and “Butter” from the recipes dataset to a variable named “sugar_butter”. Check for the “type” attribute values in the dataset and if it matches “Muffin” then assign 0 otherwise assign 1 to the “type_label” variable.

Step5: Import the “SVC” module from “svm” library and set the “kernel” type to linear. Fit the model using “sugar_butter” and “type_label” as the parameters.

Step6: Use the “model.coef” function to get the coefficients of the linear model trained in above steps and assign the values to a list named “w”. Now assign variable “a” with value equal to “-w[0]/w[1]”. Using the “linespace()” function to equally space instances from 5 to 30 and then assign it to variable “xx”. Now assign variable “y” the value equivalent to “a” times “xx” minus the first value of the model intercept divided by “w[1]”.

Step7: Assign the first support vector to the variable “b”. Assign the variable “yy_down” value equal to “a” times “xx” plus the “b[1]” minus “a” times “b[0]”. Assign the variable “b” the last support vector and repeat the same steps.

Step8: Using the “sns.lmplot()” function where the inputs are “Sugar” and “Flour”. Assign “recipes” to “data”, “Type” to “hue”, “Set1” to “palette”,”False” to “fit_reg” and “{“s”:70}” to “sctter_kws” and plot the graph with “xx” and “yy” as the parameters.

Step9: Plot the same graph in “Step8” along with a line with “xx”,”yy_down” and “ ‘k–-’ ” as the parameters, a graph with “xx”,”yy_down” and “ ‘k–-’ ” as the parameters and a scatter plot with first and last support vectors.

Step10: Import “confusion_matrix” and “classification_report”from “sklearn.metrics” and ”train_test_split” from “sklearn.model_selection”.

Step11: Assign “x_train”,”x_test”,”y_train” and “y_test” the train_test_split with a ratio of 0.2.

Step12: Fit a new model named “model1” with the partial dataset.

Step13: Using the “predict()” function on “model1” with “x_test” as the parameter, assign the value to variable “pred”.

Step14: Display the confusion matrix and the classification report.

IMPLEMENTATION:-

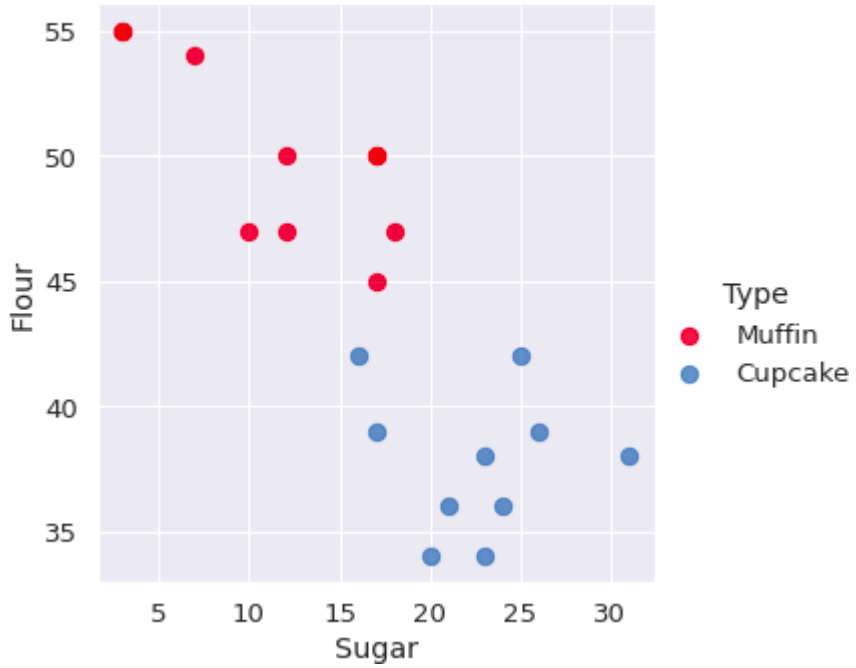
```
import numpy as np
import pandas as pd
from sklearn import svm
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(font_scale=1.2)

recipes=pd.read_csv('../input/muffins-datset/recipes_muffins_cupcakes.csv')
recipes.head()
recipes.shape
```

```
(20, 9)
```

```
sns.lmplot('Sugar','Flour',data=recipes,hue='Type',palette='Set1',fit_reg=False,scatter_kws={"s":70})
```

```
<seaborn.axisgrid.FacetGrid at 0x7fca4a9fda90>
```




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

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

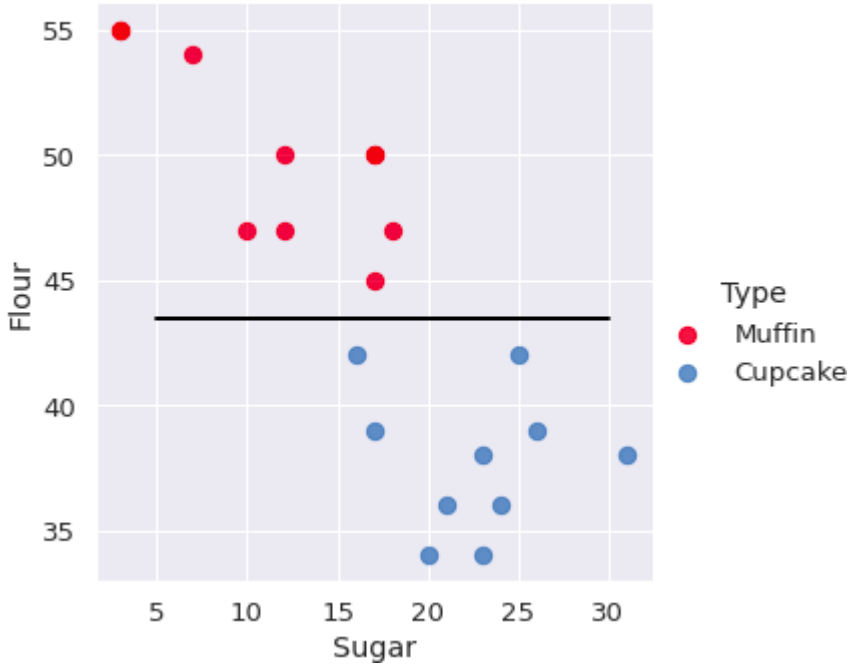
```
SVC(kernel='linear')
```

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

b=model.support_vectors_[0] #plot to seperate hyperplane that pass
yy_down=a*xx+(b[1]-a*b[0])
b=model.support_vectors_[-1
] yy_up=a*xx+(b[1]-a*b[0])
```

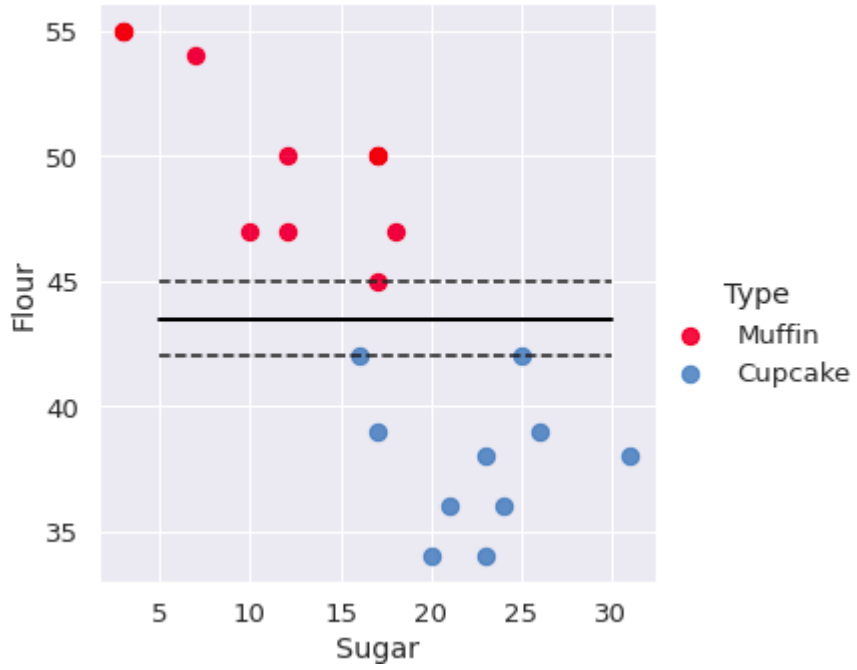
```
sns.lmplot('Sugar','Flour',data=recipes,hue='Type',palette='Set1',fit_reg=False,scatter_kws={"s":70})
plt.plot(xx,yy,linewidth=2,color='black')
```

```
[<matplotlib.lines.Line2D at 0x7fca4a98ba50>]
```




```
sns.lmplot('Sugar','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=80,facecol or='none')
```

```
<matplotlib.collections.PathCollection at 0x7fca4a880710>
```




```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

x_train,x_test,y_train,y_test =
train_test_split(sugar_butter,type_label,test_size=0.2)

model1=svm.SVC(kernel='linear')
model1.fit(x_train,y_train)
```

```
SVC(kernel='linear')
```

```
pred = model1.predict(x_test)
print(pred)
```

```
[0 0 1 0]
```

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

```
[[2 0]
 [1 1]]
```

print(classification_report(y_test,pred))

	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 SVM classifier model has been executed successfully and the classified output has been analyzed for the given dataset(muffins.csv).