In [1]:

```
# Importig Libraries
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.preprocessing import StandardScaler

from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report


from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score
```

In [2]:

```
# Loading Dataset
data = pd.read_csv('forestfires.csv')
```

In [3]:

```
#EDA & Data preprocessing
data.shape
```

Out[3]:

```
(517, 31)
```

In [4]:

```
data.head()
```

Out[4]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	mont
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	

5 rows × 31 columns

In [5]:

```
data.sample(10)
```

Out[5]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	r
326	sep	sat	92.2	102.3	751.5	8.4	24.1	27	3.1	0.0	...	0	0	
315	sep	wed	91.2	134.7	817.5	7.2	18.5	30	2.7	0.0	...	0	0	
189	mar	sun	90.7	44.0	92.4	5.5	11.5	60	4.0	0.0	...	0	0	
341	sep	mon	91.9	111.7	770.3	6.5	15.7	51	2.2	0.0	...	0	0	
101	aug	tue	88.8	147.3	614.5	9.0	14.4	66	5.4	0.0	...	0	0	
255	aug	thu	87.5	77.0	694.8	5.0	22.3	46	4.0	0.0	...	0	0	
225	sep	sun	93.5	149.3	728.6	8.1	22.9	39	4.9	0.0	...	0	0	
126	mar	mon	87.6	52.2	103.8	5.0	9.0	49	2.2	0.0	...	0	0	
24	aug	sat	93.5	139.4	594.2	20.3	23.7	32	5.8	0.0	...	0	0	
398	aug	sat	93.7	231.1	715.1	8.4	25.9	32	3.1	0.0	...	0	0	

10 rows × 31 columns

In [6]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517 entries, 0 to 516
Data columns (total 31 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   month          517 non-null    object 
 1   day            517 non-null    object 
 2   FFMC           517 non-null    float64
 3   DMC            517 non-null    float64
 4   DC             517 non-null    float64
 5   ISI            517 non-null    float64
 6   temp           517 non-null    float64
 7   RH             517 non-null    int64  
 8   wind           517 non-null    float64
 9   rain           517 non-null    float64
 10  area           517 non-null    float64
 11  dayfri         517 non-null    int64  
 12  daymon         517 non-null    int64  
 13  daysat         517 non-null    int64  
 14  daysun         517 non-null    int64  
 15  daythu         517 non-null    int64  
 16  daytue         517 non-null    int64  
 17  daywed         517 non-null    int64  
 18  monthapr       517 non-null    int64  
 19  monthaug       517 non-null    int64  
 20  monthdec       517 non-null    int64  
 21  monthfeb       517 non-null    int64  
 22  monthjan       517 non-null    int64  
 23  monthjul       517 non-null    int64  
 24  monthjun       517 non-null    int64  
 25  monthmar       517 non-null    int64  
 26  monthmay       517 non-null    int64  
 27  monthnov       517 non-null    int64  
 28  monthoct       517 non-null    int64  
 29  monthsep       517 non-null    int64  
 30  size_category  517 non-null    object 
dtypes: float64(8), int64(20), object(3)
memory usage: 125.3+ KB
```

In [7]:

```
data.describe()
```

Out[7]:

	FFMC	DMC	DC	ISI	temp	RH	wind	
count	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	517.000000	51
mean	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201	4.017602	
std	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469	1.791653	
min	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000	0.400000	
25%	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000	2.700000	
50%	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000	4.000000	
75%	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000	4.900000	
max	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000	9.400000	

8 rows × 28 columns

In [8]:

```
data.isna().sum()
```

Out[8]:

```
month            0
day              0
FFMC             0
DMC              0
DC               0
ISI              0
temp             0
RH               0
wind             0
rain             0
area             0
dayfri           0
daymon           0
daysat           0
daysun           0
daythu           0
daytue           0
daywed           0
monthapr         0
monthaug         0
monthdec         0
monthfeb         0
monthjan         0
monthjul         0
monthjun         0
monthmar         0
monthmay         0
monthnov         0
monthoct         0
monthsep         0
size_category    0
dtype: int64
```

In [9]:

```
# Dropping columns which are not required

data = data.drop(['dayfri', 'daymon', 'daysat', 'daysun', 'daythu','daytue', 'daywed', 'mon
                  'monthfeb','monthjan', 'monthjul', 'monthjun', 'monthmar', 'monthmay', 'm
                 axis = 1)
data
```

Out[9]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	size_category
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	small
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	small
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	small
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	small
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	small
...	...	...	...	...	...	...	...	...	...	...	...	...
512	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	large
513	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	large
514	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	large
515	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	small
516	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	small

517 rows × 12 columns

In [10]:

```
# Checking how much datapoints are having small and large area
data.size_category.value_counts()
```

Out[10]:

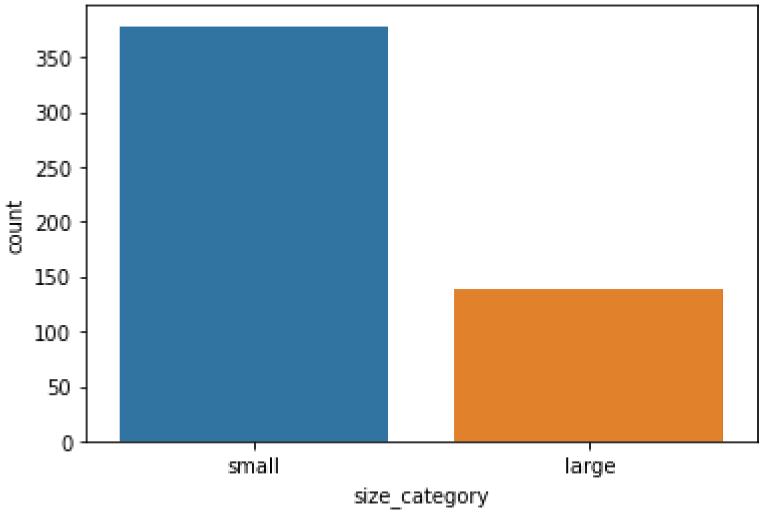
```
small    378
large    139
Name: size_category, dtype: int64
```

In [11]:

```
import seaborn as sns
sns.countplot(x = 'size_category', data = data)
```

Out[11]:

```
<AxesSubplot:xlabel='size_category', ylabel='count'>
```



In [12]:

```
# Checking for which value of area is categorised into large and small by creating crosstab
pd.crosstab(data.area, data.size_category)
```

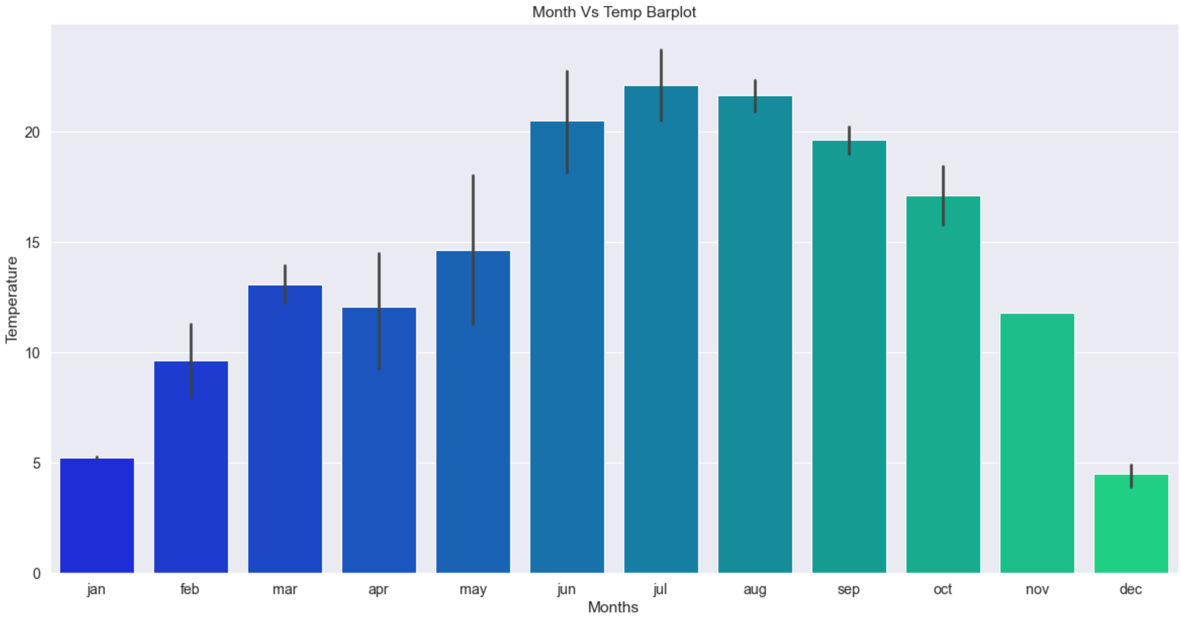
Out[12]:

size_category	**large**	**small**
area		
0.00	0	247
0.09	0	1
0.17	0	1
0.21	0	1
0.24	0	1
...	...	...
200.94	1	0
212.88	1	0
278.53	1	0
746.28	1	0
1090.84	1	0

251 rows × 2 columns

```
In [13]:
```

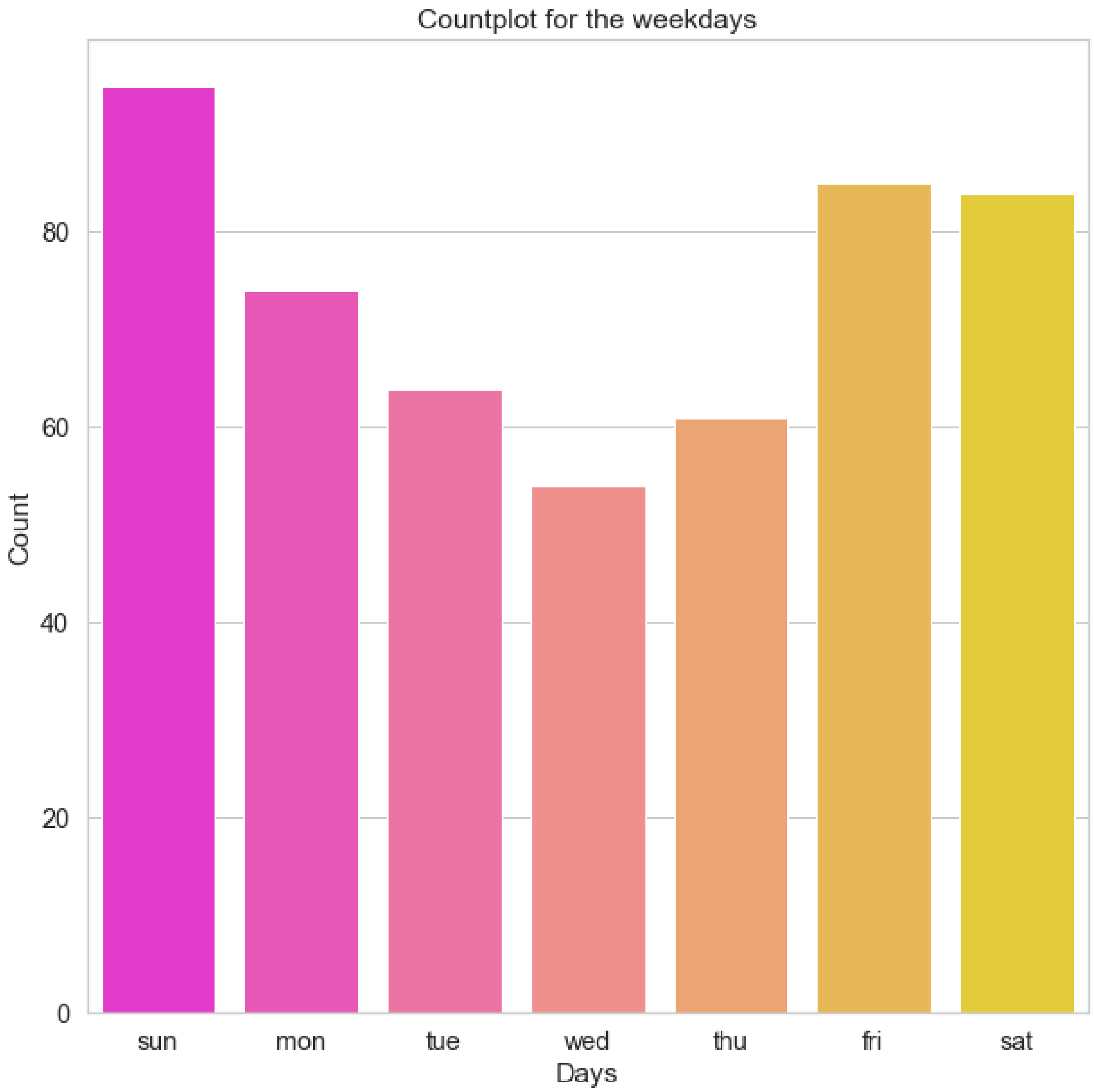
```
# Plotting Month Vs. temp plot
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = [20, 10]
sns.set(style = "darkgrid", font_scale = 1.3)
month_temp = sns.barplot(x = 'month', y = 'temp', data = data,
                         order = ['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', '
month_temp.set(title = "Month Vs Temp Barplot", xlabel = "Months", ylabel = "Temperature");
```



In [14]:

```
plt.rcParams['figure.figsize'] = [10, 10]
sns.set(style = 'whitegrid', font_scale = 1.3)
day = sns.countplot(data['day'], order = ['sun' ,'mon', 'tue', 'wed', 'thu', 'fri', 'sat'],
day.set(title = 'Countplot for the weekdays', xlabel = 'Days', ylabel = 'Count');
```

```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\_decorators.py:3
6: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinter
pretation.
  warnings.warn(
```

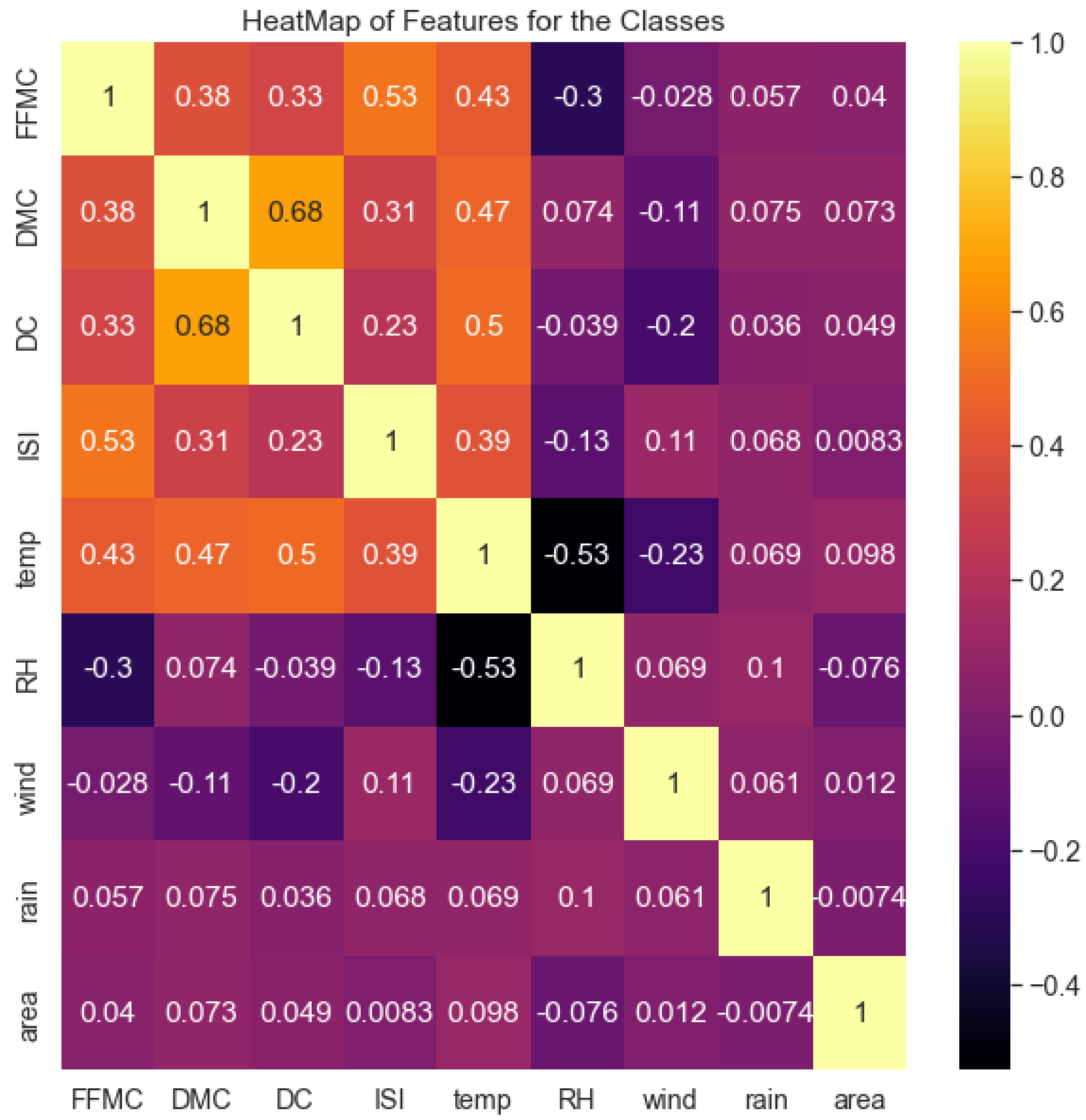


In [15]:

```
sns.heatmap(data.corr(), annot=True, cmap="inferno")
ax = plt.gca()
ax.set_title("HeatMap of Features for the Classes")
```

Out[15]:

```
Text(0.5, 1.0, 'HeatMap of Features for the Classes')
```



In [16]:

```
data.head()
```

Out[16]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	size_category
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	small
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	small
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	small
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	small
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	small

In [17]:

```
# Encoding month and day features

data.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'
                           (1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)
data.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7), inplace=True)
data.head()
```

Out[17]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	size_category
0	3	5	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	small
1	10	2	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	small
2	10	6	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	small
3	3	5	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	small
4	3	7	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	small

In [18]:

```
# Encoding target variable 'size category'

data.size_category.replace(('small', 'large'), (0, 1), inplace = True)
data.sample(5)
```

Out[18]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	size_category
444	9	5	90.3	290.0	855.3	7.4	16.2	58	3.6	0.0	9.96	1
393	3	2	93.4	15.0	25.6	11.4	15.2	19	7.6	0.0	0.00	0
512	8	7	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	1
450	8	3	95.2	217.7	690.0	18.0	23.4	49	5.4	0.0	6.43	1
348	9	5	92.1	99.0	745.3	9.6	17.4	57	4.5	0.0	0.00	0

In [19]:

```
data.corr()['size_category'].sort_values(ascending=False)
```

Out[19]:

```
size_category    1.000000
area             0.311322
month            0.080316
wind             0.059113
rain             0.050001
DMC              0.034715
FFMC             0.022063
DC               0.019428
day              0.016796
temp             0.006021
ISI             -0.008726
RH              -0.045243
Name: size_category, dtype: float64
```

In [20]:

```
# Standardizing data

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()

scaler.fit(data.drop('size_category',axis=1))
```

Out[20]:

```
▾ StandardScaler
StandardScaler()
```

In [21]:

```
scaled_features=scaler.transform(data.drop('size_category',axis=1))
data_head=pd.DataFrame(scaled_features,columns=data.columns[:-1])
data_head
```

Out[21]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	
0	-1.968443	0.357721	-0.805959	-1.323326	-1.830477	-0.860946	-1.842640	0.411724	1.4
1	1.110120	-1.090909	-0.008102	-1.179541	0.488891	-0.509688	-0.153278	-0.692456	-1.7
2	1.110120	0.840597	-0.008102	-1.049822	0.560715	-0.509688	-0.739383	-0.692456	-1.5
3	-1.968443	0.357721	0.191362	-1.212361	-1.898266	-0.004756	-1.825402	3.233519	-0.0
4	-1.968443	1.323474	-0.243833	-0.931043	-1.798600	0.126966	-1.291012	3.356206	-1.2
...	...	...	...	...	...	...	...	...	
512	0.230531	1.323474	-1.640083	-0.846648	0.474768	-1.563460	1.536084	-0.753800	-0.7
513	0.230531	1.323474	-1.640083	-0.846648	0.474768	-1.563460	0.519019	1.638592	0.9
514	0.230531	1.323474	-1.640083	-0.846648	0.474768	-1.563460	0.398350	1.577248	1.4
515	0.230531	0.840597	0.680957	0.549003	0.269382	0.500176	1.156839	-0.140366	-0.0
516	1.549915	-1.090909	-2.020879	-1.685913	-1.780442	-1.739089	-1.222058	-0.815143	0.2

517 rows × 11 columns

In [22]:

```
# Splitting data into test data and train data

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(data_head,data['size_category'], test_s

print('Shape of x_train: ', x_train.shape)
print('Shape of x_test: ', x_test.shape)
print('Shape of y_train: ', y_train.shape)
print('Shape of y_test: ', y_test.shape)
```

```
Shape of x_train:  (361, 11)
Shape of x_test:  (156, 11)
Shape of y_train:  (361,)
Shape of y_test:  (156,)
```

In [23]:

```
#Building SVM model
from sklearn import metrics

svc = SVC()
svc.fit(x_train, y_train)
# make predictions
prediction = svc.predict(x_test)
# summarize the fit of the model
print(metrics.classification_report(y_test, prediction))
print(metrics.confusion_matrix(y_test, prediction))

print("Accuracy:",metrics.accuracy_score(y_test, prediction))
print("Precision:",metrics.precision_score(y_test, prediction))
print("Recall:",metrics.recall_score(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.78      0.99      0.87       115
           1       0.90      0.22      0.35        41

    accuracy                           0.79       156
   macro avg       0.84      0.61      0.61       156
weighted avg       0.81      0.79      0.74       156

[[114   1]
 [ 32   9]]
Accuracy: 0.7884615384615384
Precision: 0.9
Recall: 0.21951219512195122
```

In [24]:

```
#Building SVM model with Hyper Parameters
model = SVC(kernel='rbf',gamma=15, C=1)

model.fit(x_train, y_train)
# make predictions
prediction = model.predict(x_test)
# summarize the fit of the model
print(metrics.classification_report(y_test, prediction))
print(metrics.confusion_matrix(y_test, prediction))

print("Accuracy:",metrics.accuracy_score(y_test, prediction))
print("Precision:",metrics.precision_score(y_test, prediction))
print("Recall:",metrics.recall_score(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.75      0.99      0.85       115
           1       0.75      0.07      0.13        41

    accuracy                           0.75       156
   macro avg       0.75      0.53      0.49       156
weighted avg       0.75      0.75      0.66       156

[[114   1]
 [ 38   3]]
Accuracy: 0.75
Precision: 0.75
Recall: 0.07317073170731707
```

In [25]:

```
#Building model with Grid Search CV
final_model = SVC(C= 15, gamma = 50, kernel = 'linear')

final_model.fit(x_train, y_train)
# make predictions
prediction = final_model.predict(x_test)
# summarize the fit of the final_model
print(metrics.classification_report(y_test, prediction))
print(metrics.confusion_matrix(y_test, prediction))

print("Accuracy:",metrics.accuracy_score(y_test, prediction))
print("Precision:",metrics.precision_score(y_test, prediction))
print("Recall:",metrics.recall_score(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       115
           1       0.95      0.95      0.95        41

    accuracy                           0.97       156
   macro avg       0.97      0.97      0.97       156
weighted avg       0.97      0.97      0.97       156

[[113   2]
 [  2  39]]
Accuracy: 0.9743589743589743
Precision: 0.9512195121951219
Recall: 0.9512195121951219
```

In [ ]:

```

```