In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.svm import SVC

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('forestfires.csv')
data
```

Out[2]:

	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
512	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	...	0	0	
513	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	...	0	0	
514	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	...	0	0	
515	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	...	0	0	
516	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	...	0	0	

517 rows × 31 columns

In [3]:
```
data.shape
```

Out[3]:
```
(517, 31)
```

In [4]:
```
data.isna().sum()
```

Out[4]:
```
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 [5]:
```
data.dtypes
```

Out[5]:
```
month             object
day               object
FFMC             float64
DMC              float64
DC               float64
ISI              float64
temp             float64
RH                 int64
wind             float64
rain             float64
area             float64
dayfri             int64
daymon             int64
daysat             int64
daysun             int64
daythu             int64
daytue             int64
daywed             int64
monthapr           int64
monthaug           int64
monthdec           int64
monthfeb           int64
monthjan           int64
monthjul           int64
monthjun           int64
monthmar           int64
monthmay           int64
monthnov           int64
monthoct           int64
monthsep           int64
size_category     object
dtype: object
```

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	517.00
mean	90.644681	110.872340	547.940039	9.021663	18.889168	44.288201	4.017602	0.02
std	5.520111	64.046482	248.066192	4.559477	5.806625	16.317469	1.791653	0.29
min	18.700000	1.100000	7.900000	0.000000	2.200000	15.000000	0.400000	0.00
25%	90.200000	68.600000	437.700000	6.500000	15.500000	33.000000	2.700000	0.00
50%	91.600000	108.300000	664.200000	8.400000	19.300000	42.000000	4.000000	0.00
75%	92.900000	142.400000	713.900000	10.800000	22.800000	53.000000	4.900000	0.00
max	96.200000	291.300000	860.600000	56.100000	33.300000	100.000000	9.400000	6.40

8 rows × 28 columns

In [8]:
```
le = LabelEncoder()
data['month'] = le.fit_transform(data['month'])
data['day'] = le.fit_transform(data['day'])
data['size_category'] = le.fit_transform(data['size_category'])
data.head()
```

Out[8]:

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

5 rows × 31 columns

In [13]:
```
data = data.drop(['dayfri', 'daymon', 'daysat', 'daysun', 'daythu','daytue', 'day
                  'monthaug', 'monthdec', 'monthfeb','monthjan', 'monthjul', 'mon
                  'monthmay', 'monthnov','monthoct','monthsep'], axis = 1)
```

In [14]:
```
data.head()
```

Out[14]:

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

In [16]:
```
std = StandardScaler()
X = data.drop('size_category',axis = 1)
std = std.fit_transform(X)
X_scaled = pd.DataFrame(data = std,columns=X.columns)
X_scaled
```

Out[16]:

	month	day	FFMC	DMC	DC	ISI	temp	RH	wir
0	0.284222	-1.423121	-0.805959	-1.323326	-1.830477	-0.860946	-1.842640	0.411724	1.4986
1	0.970871	1.176715	-0.008102	-1.179541	0.488891	-0.509688	-0.153278	-0.692456	-1.7417
2	0.970871	-0.383187	-0.008102	-1.049822	0.560715	-0.509688	-0.739383	-0.692456	-1.5182
3	0.284222	-1.423121	0.191362	-1.212361	-1.898266	-0.004756	-1.825402	3.233519	-0.0098
4	0.284222	0.136781	-0.243833	-0.931043	-1.798600	0.126966	-1.291012	3.356206	-1.2389
...	...	...	...	...	...	...	...	...	
512	-1.089076	0.136781	-1.640083	-0.846648	0.474768	-1.563460	1.536084	-0.753800	-0.7361
513	-1.089076	0.136781	-1.640083	-0.846648	0.474768	-1.563460	0.519019	1.638592	0.9957
514	-1.089076	0.136781	-1.640083	-0.846648	0.474768	-1.563460	0.398350	1.577248	1.4986
515	-1.089076	-0.383187	0.680957	0.549003	0.269382	0.500176	1.156839	-0.140366	-0.0098
516	0.741988	1.176715	-2.020879	-1.685913	-1.780442	-1.739089	-1.222058	-0.815143	0.2695

517 rows × 11 columns

In [17]:
```
X = X_scaled
y = data[['size_category']]
```

In [19]:
```
t,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=12,stratify=y
```

In [20]:
```
X_train.shape,y_train.shape
```

Out[20]: ((413, 11), (413, 1))

In [21]:
```
X_test.shape,y_test.shape
```

Out[21]: ((104, 11), (104, 1))

In [22]:
```
model_linear = SVC(kernel = 'linear',C=30)
model_linear.fit(X_train,y_train)
```

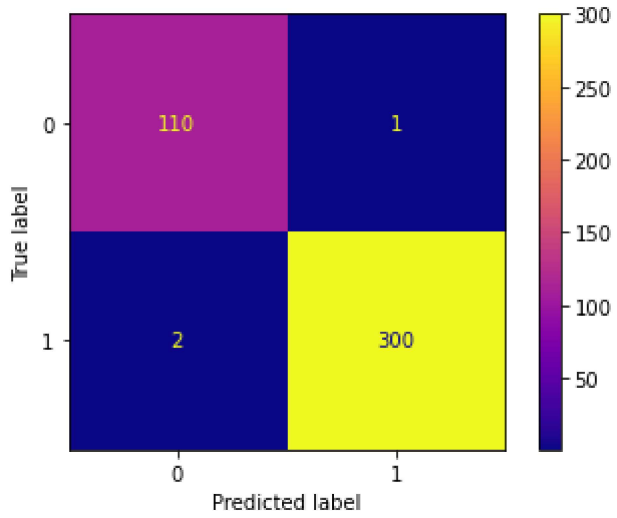
Out[22]: SVC(C=30, kernel='linear')

In [23]:
```
y_pred_train_lr = model_linear.predict(X_train)
y_pred_test_lr = model_linear.predict(X_test)
```

In [24]:
```
t('Accuracy Score :',accuracy_score(y_train,y_pred_train_lr))
t('\n Confusion Matrix : \n ',confusion_matrix(y_train,y_pred_train_lr))
t('\n Classification Report :\n ',classification_report(y_train,y_pred_train_lr))
```

```
Accuracy Score : 0.9927360774818402

 Confusion Matrix :
  [[110   1]
 [  2 300]]

 Classification Report :
                precision    recall  f1-score   support

           0       0.98      0.99      0.99       111
           1       1.00      0.99      1.00       302

    accuracy                           0.99       413
   macro avg       0.99      0.99      0.99       413
weighted avg       0.99      0.99      0.99       413
```

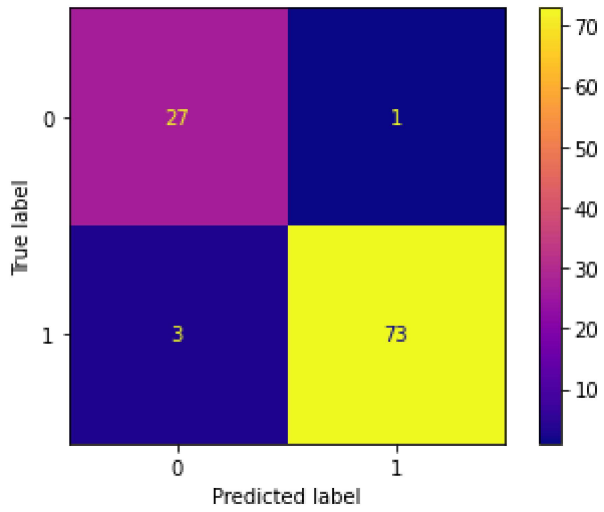
In [25]:
```
plot_confusion_matrix(model_linear, X_train, y_train, cmap='plasma')
plt.show()
```



In [26]:
```
print('Accuracy Score :',accuracy_score(y_test,y_pred_test_lr))
print('\n Confusion Matrix : \n ',confusion_matrix(y_test,y_pred_test_lr))
print('\n Classification Report :\n ',classification_report(y_test,y_pred_test_lr
```

```
Accuracy Score : 0.9615384615384616

 Confusion Matrix :
  [[27  1]
 [ 3 73]]

 Classification Report :
               precision    recall  f1-score   support

           0       0.90      0.96      0.93        28
           1       0.99      0.96      0.97        76

    accuracy                           0.96       104
   macro avg       0.94      0.96      0.95       104
weighted avg       0.96      0.96      0.96       104
```

In [27]:
```
plot_confusion_matrix(model_linear, X_test, y_test, cmap='plasma')
plt.show()
```



In [28]:
```
model_rbf = SVC(kernel='rbf',C=30,gamma=0.1)
model_rbf.fit(X_train,y_train)
```

Out[28]:
```
SVC(C=30, gamma=0.1)
```

In [31]:
```
y_pred_train_rbf = model_rbf.predict(X_train)
y_pred_test_rbf = model_rbf.predict(X_test)
```

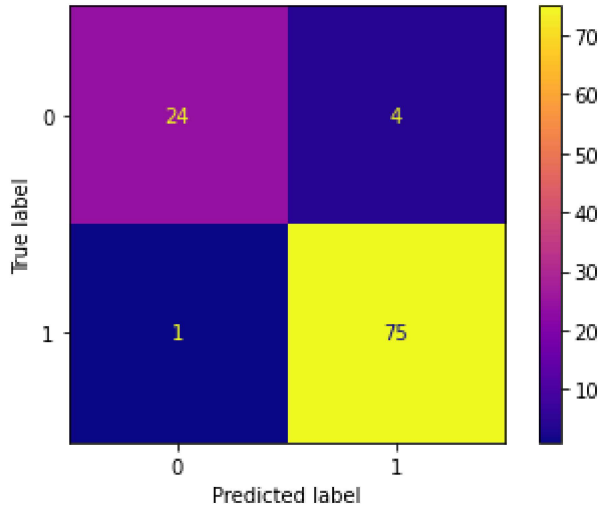
In [33]:
```
print('Accuracy score :',accuracy_score(y_train,y_pred_train_rbf))
print('Confision matrix :\n',confusion_matrix(y_train,y_pred_train_rbf))
print('Classification report :\n',classification_report(y_train,y_pred_train_rbf))
```

```
Accuracy score : 0.9757869249394673
Confision matrix :
 [[102   9]
 [  1 301]]
Classification report :
               precision    recall  f1-score   support

           0       0.99      0.92      0.95       111
           1       0.97      1.00      0.98       302

    accuracy                           0.98       413
   macro avg       0.98      0.96      0.97       413
weighted avg       0.98      0.98      0.98       413
```

In [35]:
```
print('Accuracy score :',accuracy_score(y_test,y_pred_test_rbf))
print('Confision matrix :\n',confusion_matrix(y_test,y_pred_test_rbf))
print('Classification report :\n',classification_report(y_test,y_pred_test_rbf))
```

```
Accuracy score : 0.9519230769230769
Confision matrix :
 [[24  4]
 [ 1 75]]
Classification report :
               precision    recall  f1-score   support

           0       0.96      0.86      0.91        28
           1       0.95      0.99      0.97        76

    accuracy                           0.95       104
   macro avg       0.95      0.92      0.94       104
weighted avg       0.95      0.95      0.95       104
```

In [36]:
```
plot_confusion_matrix(model_rbf, X_test, y_test, cmap='plasma')
plt.show()
```



In [44]:
```
model_poly = SVC(kernel='poly')
model_poly.fit(X_train,y_train)
```

Out[44]:
```
SVC(kernel='poly')
```

In [45]:
```
y_pred_poly_train = model_poly.predict(X_train)
y_pred_poly_test = model_poly.predict(X_test)
```

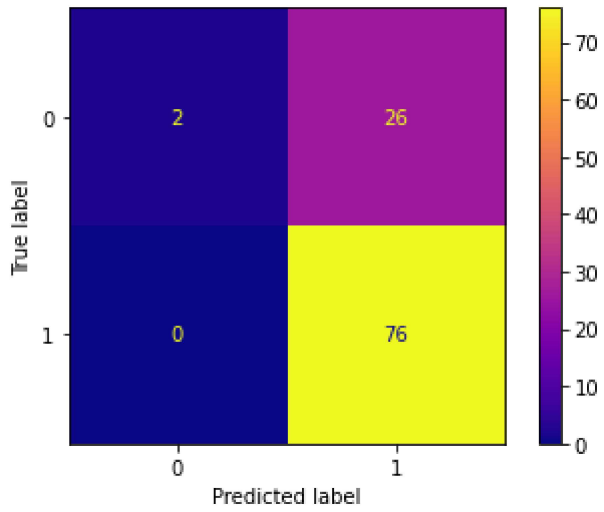
In [46]:
```
print('Accuracy Score :',accuracy_score(y_train,y_pred_poly_train))
print('\n Confusion Matrix : \n ',confusion_matrix(y_train,y_pred_poly_train))
print('\n Classification Report :\n ',classification_report(y_train,y_pred_poly_t
```

```
Accuracy Score : 0.8038740920096852

 Confusion Matrix :
  [[ 31  80]
 [  1 301]]

 Classification Report :
               precision    recall  f1-score   support

           0       0.97      0.28      0.43       111
           1       0.79      1.00      0.88       302

    accuracy                           0.80       413
   macro avg       0.88      0.64      0.66       413
weighted avg       0.84      0.80      0.76       413
```

In [47]:
```
t('Accuracy Score :',accuracy_score(y_test,y_pred_poly_test))
t('\n Confusion Matrix : \n ',confusion_matrix(y_test,y_pred_poly_test))
t('\n Classification Report :\n ',classification_report(y_test,y_pred_poly_test))
```

```
Accuracy Score : 0.75

 Confusion Matrix :
  [[ 2 26]
 [ 0 76]]

 Classification Report :
                precision    recall  f1-score   support

           0       1.00      0.07      0.13        28
           1       0.75      1.00      0.85        76

    accuracy                           0.75       104
   macro avg       0.87      0.54      0.49       104
weighted avg       0.81      0.75      0.66       104
```

In [48]:
```
plot_confusion_matrix(model_poly, X_test, y_test, cmap='plasma')
plt.show()
```



In [ ]:
```

```