In [41]:
```
ıs pd
; np
› import pyplot as plt
.ine
›del_selection import train_test_split
·eprocessing import LabelEncoder,StandardScaler
:trics import accuracy_score,confusion_matrix,classification_report,plot_confusior
⁄m import SVC

;
·warnings('ignore')
```

In [42]:
```
train = pd.read_csv('SalaryData_Train(1).csv')
test = pd.read_csv('SalaryData_Test(1).csv')
```

In [43]:
```
data = pd.merge(train,test,)
data.head()
```

Out[43]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | ca |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 38 | Private | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | |
| **1** | 19 | Private | HS-grad | 9 | Never-married | Craft-repair | Own-child | White | Male | |
| **2** | 19 | Private | HS-grad | 9 | Never-married | Craft-repair | Own-child | White | Male | |
| **3** | 19 | Private | HS-grad | 9 | Never-married | Craft-repair | Own-child | White | Male | |
| **4** | 19 | Private | HS-grad | 9 | Never-married | Craft-repair | Own-child | White | Male | |

In [44]:
```
data.shape
```

Out[44]:
```
(5910, 14)
```

In [45]:
```
data.isna().sum()
```

Out[45]:
```
age              0
workclass        0
education        0
educationno      0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
capitalgain      0
capitalloss      0
hoursperweek     0
native           0
Salary           0
dtype: int64
```

In [46]:
```
data.dtypes
```

Out[46]:
```
age               int64
workclass        object
education        object
educationno       int64
maritalstatus    object
occupation       object
relationship     object
race             object
sex              object
capitalgain       int64
capitalloss       int64
hoursperweek      int64
native           object
Salary           object
dtype: object
```

In [47]:
```
le = LabelEncoder()
```

In [48]:
```
data.workclass.unique()
```

Out[48]:
```
array([' Private', ' Local-gov', ' Federal-gov', ' Self-emp-inc',
       ' Self-emp-not-inc', ' State-gov'], dtype=object)
```

In [49]:
```
data['workclass'] = le.fit_transform(data['workclass'])
```

In [50]:
```
data.education.unique()
```

Out[50]:
```
array([' HS-grad', ' Some-college', ' Masters', ' Bachelors', ' 11th',
       ' 1st-4th', ' Assoc-acdm', ' Assoc-voc', ' 10th', ' 7th-8th',
       ' 9th', ' 12th', ' 5th-6th', ' Prof-school', ' Doctorate',
       ' Preschool'], dtype=object)
```

In [51]:
```
data['education'] = le.fit_transform(data['education'])
```

```
In [52]: data.maritalstatus.unique()

Out[52]: array([' Divorced', ' Never-married', ' Married-civ-spouse', ' Separated',
                ' Widowed'], dtype=object)
```

```
In [53]: data['maritalstatus'] = le.fit_transform(data['maritalstatus'])
```

```
In [54]: data.occupation.unique()

Out[54]: array([' Handlers-cleaners', ' Craft-repair', ' Machine-op-inspct',
                ' Transport-moving', ' Other-service', ' Prof-specialty',
                ' Exec-managerial', ' Sales', ' Adm-clerical', ' Tech-support',
                ' Protective-serv', ' Farming-fishing'], dtype=object)
```

```
In [55]: data['occupation'] = le.fit_transform(data['occupation'])
```

```
In [56]: data.relationship.unique()

Out[56]: array([' Not-in-family', ' Own-child', ' Husband', ' Wife', ' Unmarried',
                ' Other-relative'], dtype=object)
```

```
In [57]: data['relationship'] = le.fit_transform(data['relationship'])
```

```
In [58]: data.race.unique()

Out[58]: array([' White', ' Black', ' Asian-Pac-Islander'], dtype=object)
```

```
In [59]: data['race'] = le.fit_transform(data['race'])
```

```
In [60]: data.sex.unique()

Out[60]: array([' Male', ' Female'], dtype=object)
```

```
In [61]: data['sex'] = le.fit_transform(data['sex'])
```

```
In [62]: data.native.unique()

Out[62]: array([' United-States', ' Mexico', ' Philippines', ' Jamaica'],
               dtype=object)
```

```
In [63]: data['native'] = le.fit_transform(data['native'])
```

In [64]:

```
data.head()
```

Out[64]:

|   | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | capi |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 38 | 2 | 11 | 9 | 0 | 4 | 1 | 2 | 1 |  |
| **1** | 19 | 2 | 11 | 9 | 2 | 1 | 3 | 2 | 1 |  |
| **2** | 19 | 2 | 11 | 9 | 2 | 1 | 3 | 2 | 1 |  |
| **3** | 19 | 2 | 11 | 9 | 2 | 1 | 3 | 2 | 1 |  |
| **4** | 19 | 2 | 11 | 9 | 2 | 1 | 3 | 2 | 1 |  |

In [65]:

```
data.dtypes
```

Out[65]:

```
age                int64
workclass          int32
education          int32
educationno        int64
maritalstatus      int32
occupation         int32
relationship       int32
race               int32
sex                int32
capitalgain        int64
capitalloss        int64
hoursperweek       int64
native             int32
Salary            object
dtype: object
```

In [66]:

```
data.Salary.unique()
```

Out[66]:

```
array([' <=50K', ' >50K'], dtype=object)
```

In [67]:

```
X_train,X_test = train_test_split(data,test_size=0.25,random_state= 0)
```

In [68]:

```
X_train = X_train.iloc[:,:-1]
y_train = X_train.iloc[:,-1]
X_test =  X_test.iloc[:,:-1]
y_test = X_test.iloc[:,-1]
```

In [84]:

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

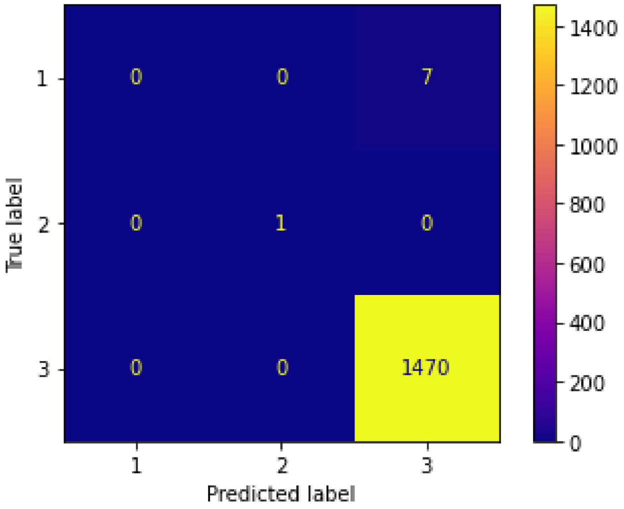
In [86]:
```
model_poly = SVC(kernel="poly")
model_poly.fit(X_train,y_train)
y_pred_poly = model_poly.predict(X_test)

np.mean(y_pred_poly == y_test)
```

Out[86]:
```
0.9952638700947226
```

In [78]:
```
print('Accuracy Score :', accuracy_score(y_test,y_pred_poly))
print('\n Confusion Matrix : \n', confusion_matrix(y_test,y_pred_poly))
print('\n Classification Report : \n', classification_report(y_test,y_pred_poly))
```

```
Accuracy Score : 0.9952638700947226

 Confusion Matrix :
 [[   0    0    7]
 [   0    1    0]
 [   0    0 1470]]

 Classification Report :
               precision    recall  f1-score   support

           1       0.00      0.00      0.00         7
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00      1470

    accuracy                           1.00      1478
   macro avg       0.67      0.67      0.67      1478
weighted avg       0.99      1.00      0.99      1478
```

In [89]:
```
plot_confusion_matrix(model_poly, X_test, y_test, cmap='plasma')
plt.show()
```



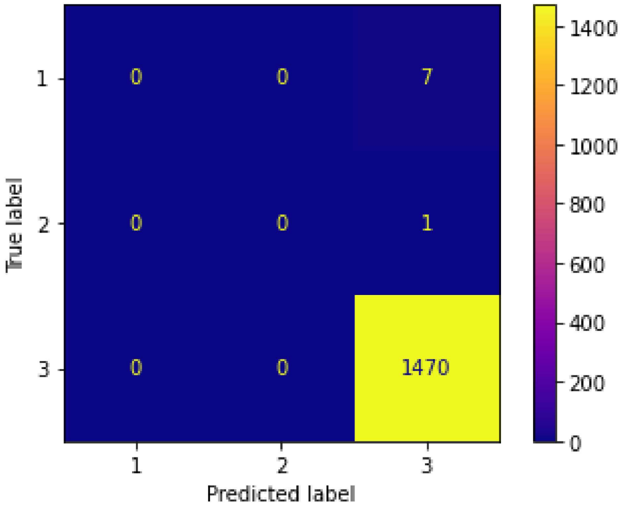
In [87]:
```
model_rbf = SVC(kernel="rbf")
model_rbf.fit(X_train,y_train)
y_pred_rbf = model_rbf.predict(X_test)

np.mean(y_pred_rbf == y_test)
```

Out[87]:
```
0.9945872801082544
```

In [82]:
```
print('Accuracy Score :', accuracy_score(y_test,y_pred_rbf))
print('\n Confusion Matrix : \n', confusion_matrix(y_test,y_pred_rbf))
print('\n Classification Report : \n', classification_report(y_test,y_pred_rbf))
```

```
Accuracy Score : 0.9945872801082544

 Confusion Matrix :
 [[   0    0    7]
 [   0    0    1]
 [   0    0 1470]]

 Classification Report :
               precision    recall  f1-score   support

           1       0.00      0.00      0.00         7
           2       0.00      0.00      0.00         1
           3       0.99      1.00      1.00      1470

    accuracy                           0.99      1478
   macro avg       0.33      0.33      0.33      1478
weighted avg       0.99      0.99      0.99      1478
```

In [90]:
```
plot_confusion_matrix(model_rbf, X_test, y_test, cmap='plasma')
plt.show()
```



In [ ]:
```
```