In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
salary_train = pd.read_csv('SalaryData_Train(1).csv')
salary_train
```

Out[2]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	s
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	M
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fem
...	...	...	...	...	...	...	...	...	
30156	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Fem
30157	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	M
30158	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Fem
30159	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	M
30160	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Fem

30161 rows × 14 columns

In [3]:
```
salary_test = pd.read_csv('SalaryData_Test(1).csv')
salary_train
```

Out[3]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	M
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fem
...	...	...	...	...	...	...	...	...	
30156	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Fem
30157	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	M
30158	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Fem
30159	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	M
30160	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Fem

30161 rows × 14 columns

In [4]:
```
#initial analysis
salary_train.shape
```

Out[4]:
```
(30161, 14)
```

In [5]:

```
salary_train.isna().sum()
```

Out[5]:

```
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 [6]:

```
salary_train.dtypes
```

Out[6]:

```
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 [7]:

```
# converting object data type to int data type.
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

In [8]:
```
salary_train['workclass'] = le.fit_transform(salary_train['workclass'])
salary_train['education'] = le.fit_transform(salary_train['education'])
salary_train['maritalstatus'] = le.fit_transform(salary_train['maritalstatus'])
salary_train['occupation'] = le.fit_transform(salary_train['occupation'])
salary_train['relationship'] = le.fit_transform(salary_train['relationship'])
salary_train['race'] = le.fit_transform(salary_train['race'])
salary_train['sex'] = le.fit_transform(salary_train['sex'])
salary_train['native'] = le.fit_transform(salary_train['native'])
salary_train['Salary'] = le.fit_transform(salary_train['Salary'])
salary_train
```

Out[8]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	39	5	9	13	4	0	1	4	1
1	50	4	9	13	2	3	0	4	1
2	38	2	11	9	0	5	1	4	1
3	53	2	1	7	2	5	0	2	1
4	28	2	9	13	2	9	5	2	0
...	...	...	...	...	...	...	...	...	...
30156	27	2	7	12	2	12	5	4	0
30157	40	2	11	9	2	6	0	4	1
30158	58	2	11	9	6	0	4	4	0
30159	22	2	11	9	4	0	3	4	1
30160	52	3	11	9	2	3	5	4	0

30161 rows × 14 columns

In [9]:
```
salary_train.dtypes
```

Out[9]:
```
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           int32
dtype: object
```

Splitting the Dataset

In [10]:

```
X = salary_train.drop(['Salary'], axis=1)
y = salary_train['Salary']
```

In [11]:

```
X
```

Out[11]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	39	5	9	13	4	0	1	4	1
1	50	4	9	13	2	3	0	4	1
2	38	2	11	9	0	5	1	4	1
3	53	2	1	7	2	5	0	2	1
4	28	2	9	13	2	9	5	2	0
...	...	...	...	...	...	...	...	...	...
30156	27	2	7	12	2	12	5	4	0
30157	40	2	11	9	2	6	0	4	1
30158	58	2	11	9	6	0	4	4	0
30159	22	2	11	9	4	0	3	4	1
30160	52	3	11	9	2	3	5	4	0

30161 rows × 13 columns

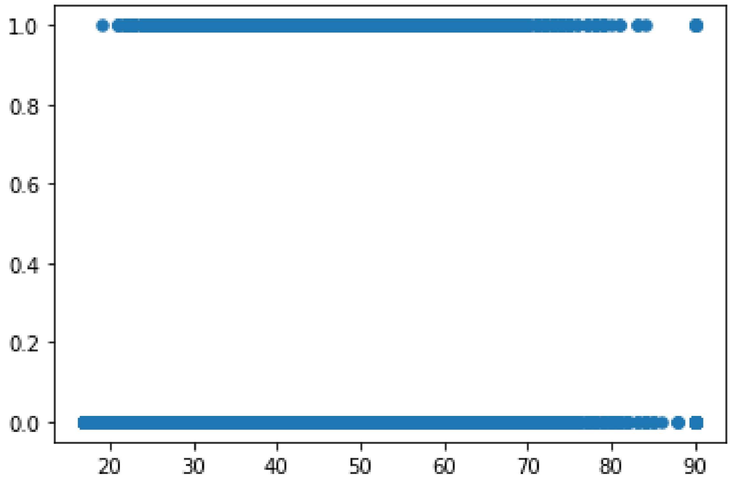
In [12]:

```
y
```

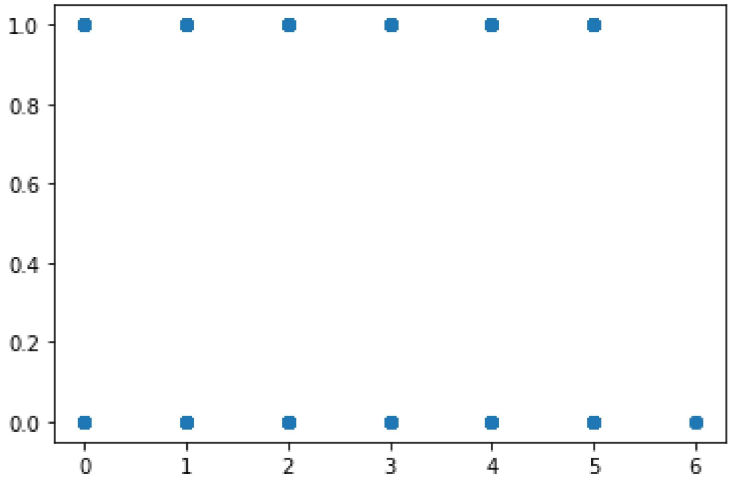
Out[12]:

```
0        0
1        0
2        0
3        0
4        0
        ..
30156    0
30157    1
30158    0
30159    0
30160    1
Name: Salary, Length: 30161, dtype: int32
```

In [13]:
```
plt.scatter(salary_train['age'], y, s=30, alpha=1)
plt.show()
```



In [14]:
```
plt.scatter(salary_train['workclass'], y, s=30, alpha=1)
plt.show()
```



Model Building

In [15]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, plot_confusion_matrix, accuracy_sc
```

In [16]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_
```

In [17]:

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[17]:

```
((24128, 13), (24128,), (6033, 13), (6033,))
```

In [18]:

```
from sklearn.svm import SVC
```

In [ ]:

```
classifier = SVC(kernel='linear', C= 1, gamma = 1)
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
```

In [ ]:

```
accuracy_score(y_test, y_test_pred)
```

In [ ]:

```
confusion_matrix(y_test, y_test_pred)
```

In [ ]:

```
plot_confusion_matrix(classifier, X_test, y_test, cmap='plasma')
plt.show()
```

In [ ]: