Assignment-17-Support_Vector_Machines-02-Salary_data

1) Prepare a classification model using SVM for salary data

Data Description:

age -- age of a person

workclass -- A work class is a grouping of work

education -- Education of an individuals

maritalstatus -- Marital status of an individulas

occupation -- occupation of an individuals

relationship -- race -- Race of an Individual sex -- Gender of an Individual

capitalgain -- profit received from the sale of an investment

capitalloss -- A decrease in the value of a capital asset

hoursperweek -- number of hours work per week

native -- Native of an individual

Salary -- salary of an individual

In [1]:

```
# SVM Classification
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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]:

```
salary_train = pd.read_csv('C:/Users/LENOVO/Documents/assignment/SalaryData_Train(1).csv')
salary_train
```

Out[2]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capita
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	
...	...	...	...	...	...	...	...	...	...	
30156	27	Private	Assoc-	12	Married-civ-	Tech-	Wife	White	Female	

In [3]:

```
salary_test = pd.read_csv('C:/Users/LENOVO/Documents/assignment/SalaryData_Test(1).csv')
salary_test
```

Out[3]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White
...	...	...	...	...	...	...	...	...
15055	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White
15056	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
15057	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
15058	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
15059	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

15060 rows × 14 columns

In [4]:

```
salary_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30161 entries, 0 to 30160
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            30161 non-null  int64 
 1   workclass      30161 non-null  object
 2   education      30161 non-null  object
 3   educationno    30161 non-null  int64 
 4   maritalstatus  30161 non-null  object
 5   occupation     30161 non-null  object
 6   relationship   30161 non-null  object
 7   race           30161 non-null  object
 8   sex            30161 non-null  object
 9   capitalgain    30161 non-null  int64 
 10  capitalloss    30161 non-null  int64 
 11  hoursperweek   30161 non-null  int64 
 12  native         30161 non-null  object
 13  Salary         30161 non-null  object
dtypes: int64(5), object(9)
memory usage: 3.2+ MB
```

In [6]:

```
salary_train.shape
```

Out[6]:

```
(30161, 14)
```

In [7]:

```
salary_test.shape
```

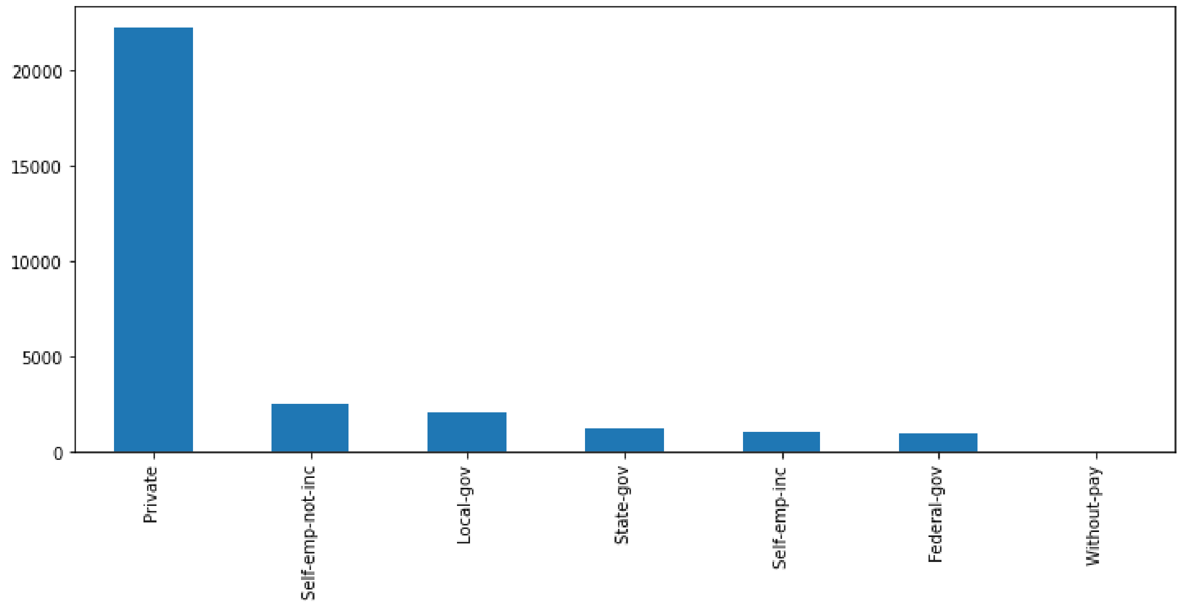
Out[7]:

```
(15060, 14)
```

Let's Visualize the data for better understanding

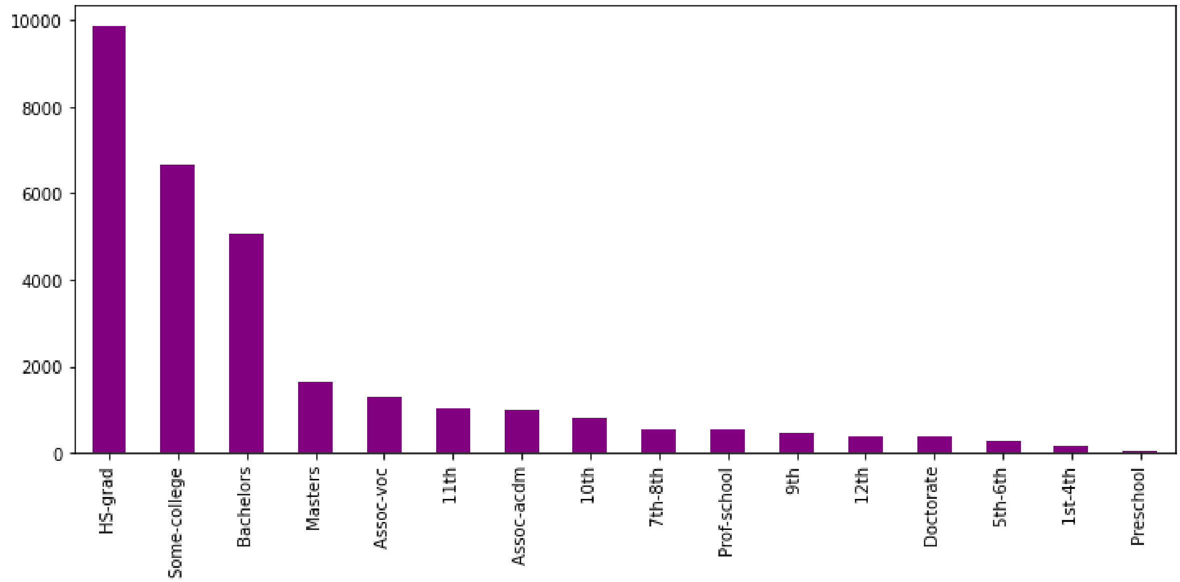
In [8]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,5))
salary_train.workclass.value_counts().plot.bar();
```



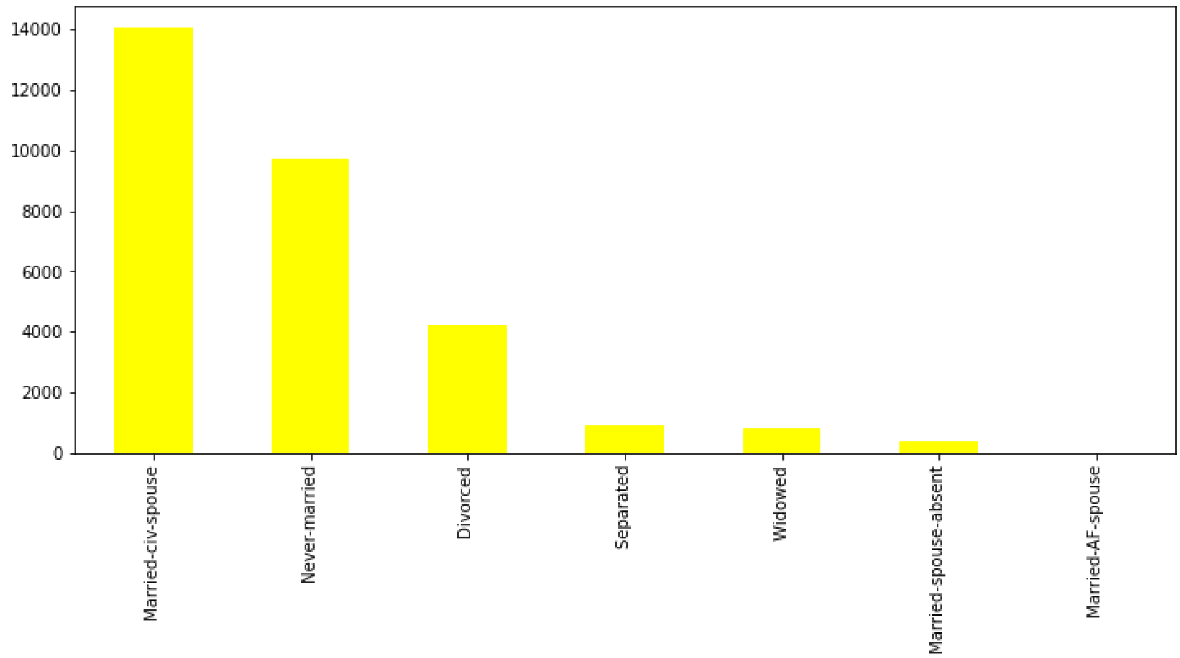
In [9]:

```
plt.figure(figsize=(12,5))
salary_train.education.value_counts().plot.bar(color='purple');
```



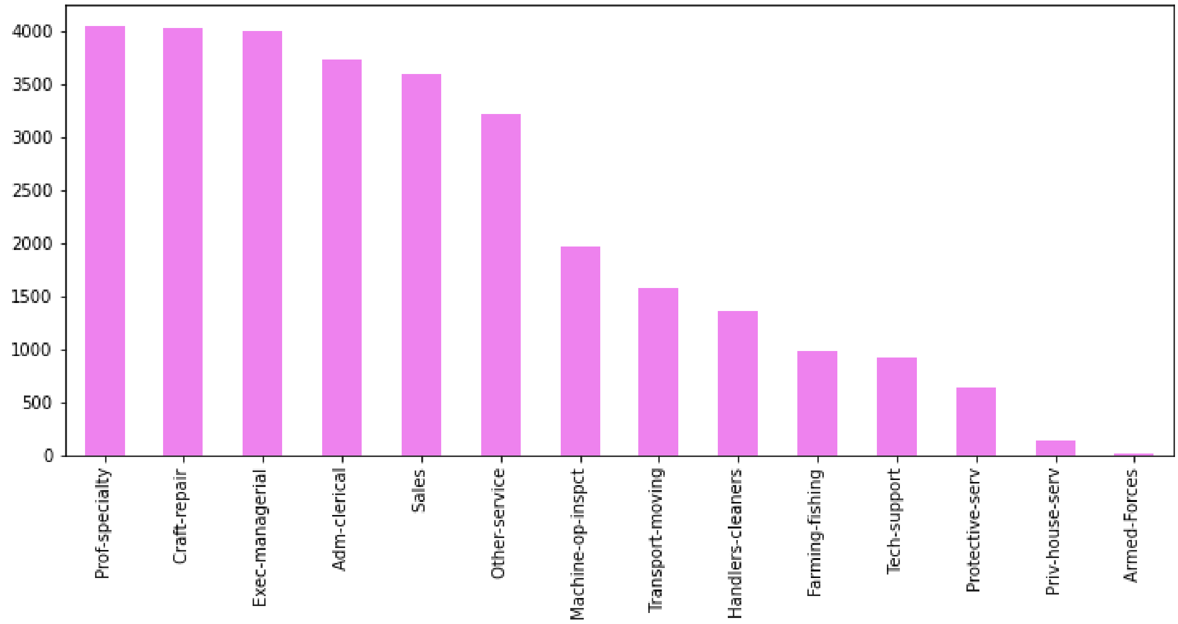
In [10]:

```
plt.figure(figsize=(12,5))
salary_train.maritalstatus.value_counts().plot.bar(color='yellow');
```



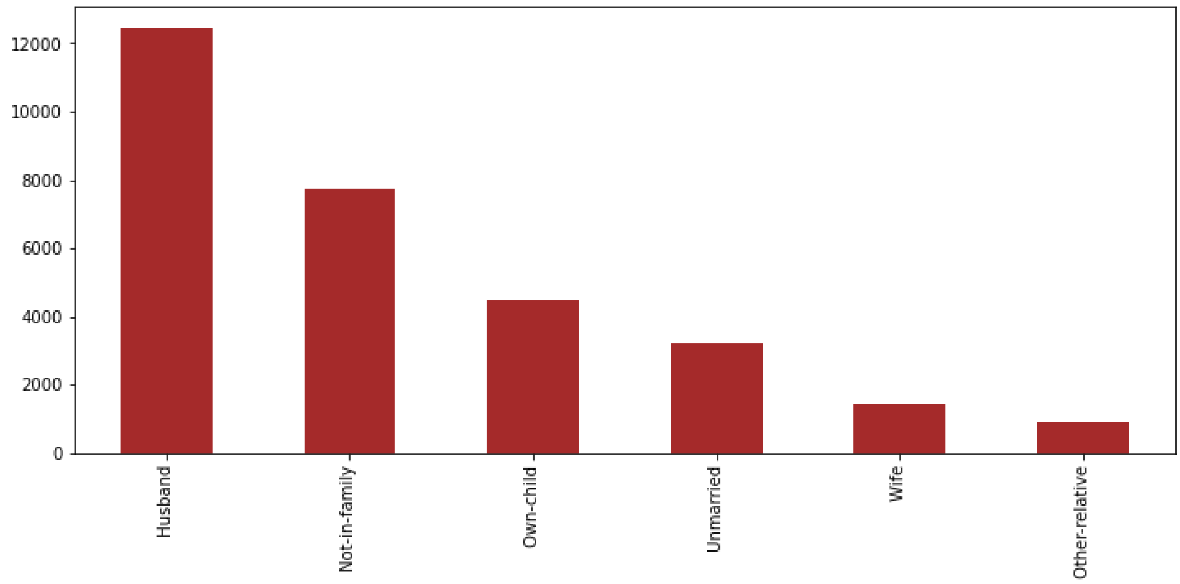
In [11]:

```
plt.figure(figsize=(12,5))
salary_train.occupation.value_counts().plot.bar(color='violet');
```



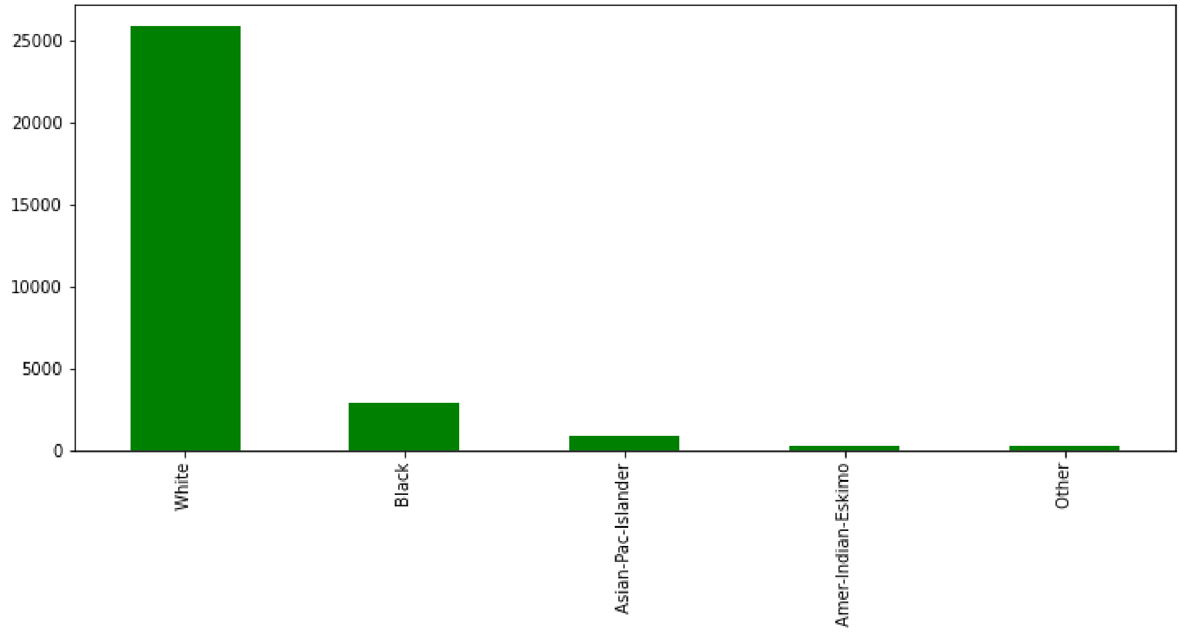
In [12]:

```
plt.figure(figsize=(12,5))
salary_train.relationship.value_counts().plot.bar(color='brown');
```



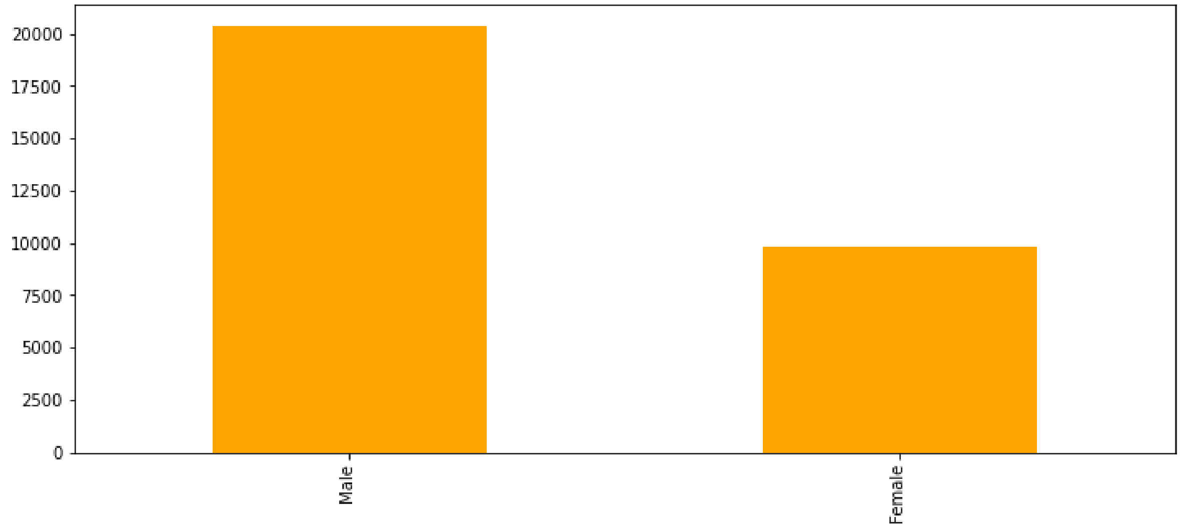
In [13]:

```
plt.figure(figsize=(12,5))
salary_train.race.value_counts().plot.bar(color='green');
```



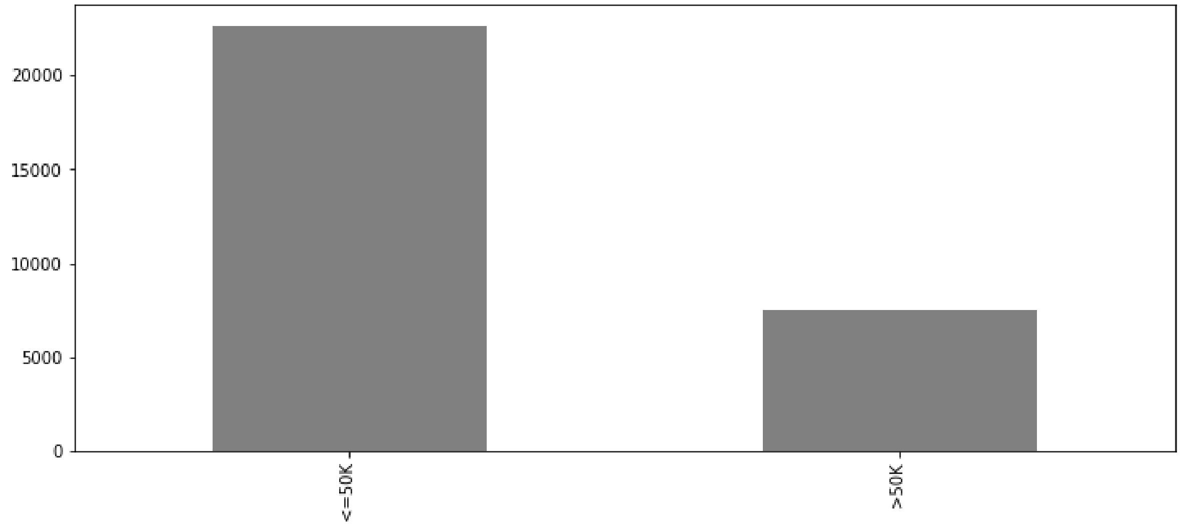
In [14]:

```
plt.figure(figsize=(12,5))
salary_train.sex.value_counts().plot.bar(color='orange');
```



In [15]:

```
plt.figure(figsize=(12,5))
salary_train.Salary.value_counts().plot.bar(color='gray');
```



In [16]:

```
# Since the Salary column is Y variable here, seperating it from the data set and applying

# For train data set
salary_train1 = salary_train.iloc[:,0:13]

salary_train1 = pd.get_dummies(salary_train1)
salary_train1
```

Out[16]:

	age	educationno	capitalgain	capitalloss	hoursperweek	workclass_Federal-gov	workclass_Local-gov	work F
0	39	13	2174	0	40	0	0	
1	50	13	0	0	13	0	0	
2	38	9	0	0	40	0	0	
3	53	7	0	0	40	0	0	
4	28	13	0	0	40	0	0	
...	...	...	...	...	...	...	...	
30156	27	12	0	0	38	0	0	
30157	40	9	0	0	40	0	0	
30158	58	9	0	0	40	0	0	
30159	22	9	0	0	20	0	0	
30160	52	9	15024	0	40	0	0	

30161 rows × 102 columns

In [17]:

```
# For test data set
salary_test1 = salary_test.iloc[:,0:13]

salary_test1 = pd.get_dummies(salary_test1)
salary_test1
```

Out[17]:

	age	educationno	capitalgain	capitalloss	hoursperweek	workclass_ Federal-gov	workclass_ Local-gov	work F
0	25	7	0	0	40	0	0	
1	38	9	0	0	50	0	0	
2	28	12	0	0	40	0	1	
3	44	10	7688	0	40	0	0	
4	34	6	0	0	30	0	0	
...	...	...	...	...	...	...	...	
15055	33	13	0	0	40	0	0	
15056	39	13	0	0	36	0	0	
15057	38	13	0	0	50	0	0	
15058	44	13	5455	0	40	0	0	
15059	35	13	0	0	60	0	0	

15060 rows × 102 columns

PCA needs to apply here as the no. of columns are more

Applyting Dimentionality Reduction technique PCA

In [18]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

#Scaling the train dataset

sc.fit(salary_train1)
salary_train_norm = sc.transform(salary_train1)
salary_train_norm                      #Normalised dataset
```

Out[18]:

```
array([[ 0.04277892,  1.12889813,  0.14608503, ...,  0.31081205,
        -0.04611353, -0.0230384 ],
       [ 0.88026081,  1.12889813, -0.14744712, ...,  0.31081205,
        -0.04611353, -0.0230384 ],
       [-0.0333558 , -0.4397325 , -0.14744712, ...,  0.31081205,
        -0.04611353, -0.0230384 ],
       ...,
       [ 1.48933854, -0.4397325 , -0.14744712, ...,  0.31081205,
        -0.04611353, -0.0230384 ],
       [-1.25151126, -0.4397325 , -0.14744712, ...,  0.31081205,
        -0.04611353, -0.0230384 ],
       [ 1.03253024, -0.4397325 ,  1.88108414, ...,  0.31081205,
        -0.04611353, -0.0230384 ]])
```

In [19]:

```
#Scaling the test dataset

sc.fit(salary_test1)
salary_test_norm = sc.transform(salary_test1)
salary_test_norm                      #Normalised dataset
```

Out[19]:

```
array([[-1.02900513, -1.2165628 , -0.14543845, ...,  0.30373366,
        -0.03554172, -0.02156441],
       [-0.05742253, -0.43489824, -0.14543845, ...,  0.30373366,
        -0.03554172, -0.02156441],
       [-0.80479376,  0.73759862, -0.14543845, ...,  0.30373366,
        -0.03554172, -0.02156441],
       ...,
       [-0.05742253,  1.1284309 , -0.14543845, ...,  0.30373366,
        -0.03554172, -0.02156441],
       [ 0.39100021,  1.1284309 ,  0.562734  , ...,  0.30373366,
        -0.03554172, -0.02156441],
       [-0.2816339 ,  1.1284309 , -0.14543845, ...,  0.30373366,
        -0.03554172, -0.02156441]])
```

In [20]:

```
from sklearn.decomposition import PCA

# For train dataset

salary_train_pca = PCA(n_components = 102)
salary_train_pca_values = salary_train_pca.fit_transform(salary_train_norm)
salary_train_pca_values
```

Out[20]:

```
array([[-5.50838008e-01, -2.38164986e+00, -5.91921169e-01, ...,
         2.83340304e-15, -2.85424188e-16,  7.46658048e-16],
       [ 2.81915829e+00, -1.37085459e+00, -4.81126421e-02, ...,
        -5.87101761e-15, -1.20725670e-15, -2.91310096e-16],
       [-7.93831525e-01,  8.71803957e-01, -1.20213150e+00, ...,
         8.07886575e-16, -9.86662354e-17, -2.00344501e-16],
       ...,
       [-2.37835145e+00, -7.98690413e-01,  3.39105780e-01, ...,
        -1.06168468e-16,  2.61345216e-16,  1.58858790e-17],
       [-1.97547719e+00,  1.19305162e+00, -1.82899406e+00, ...,
        -9.25756848e-17, -1.21746295e-17, -2.24094416e-17],
       [ 7.62131786e-01, -1.77200870e+00,  5.36971989e-01, ...,
        -4.39514634e-16,  2.30375245e-16,  1.00386091e-17]])
```

In [21]:

```
# For test dataset

salary_test_pca = PCA(n_components = 102)
salary_test_pca_values = salary_test_pca.fit_transform(salary_test_norm)
salary_test_pca_values
```

Out[21]:

```
array([[-2.24293780e+00,  2.60318091e+00, -3.27616503e-01, ...,
         5.46944667e-16, -3.47408014e-15, -3.25393797e-15],
       [ 2.22690391e+00,  1.59471521e+00, -7.32082794e-01, ...,
         3.42148515e-15,  7.79620529e-15,  7.72050169e-15],
       [ 2.30704416e+00, -1.16883181e+00, -2.00521481e-01, ...,
        -1.72555027e-16, -7.46056044e-16,  5.28633021e-16],
       ...,
       [ 2.39583218e+00, -1.46859740e+00, -2.09424792e-01, ...,
        -1.30180758e-16, -1.32788907e-16,  2.25635711e-17],
       [-1.14039506e+00, -1.03678137e+00,  2.58079490e+00, ...,
         2.24826461e-16, -2.68154165e-16,  9.86840423e-17],
       [ 3.38445120e+00, -1.95481575e+00, -1.72791531e-01, ...,
        -2.69053009e-17, -1.03121990e-17,  8.79857927e-17]])
```

In [22]:

```
# The amount of variance that each PCA explains is
var = salary_train_pca.explained_variance_ratio_
var
```

Out[22]:

```
array([4.47952203e-02, 3.03018755e-02, 2.56772664e-02, 2.30740938e-02,
       1.90544461e-02, 1.75159608e-02, 1.66112958e-02, 1.51765356e-02,
       1.40918479e-02, 1.37139289e-02, 1.30161578e-02, 1.27145892e-02,
       1.22845420e-02, 1.20633855e-02, 1.19277829e-02, 1.17776199e-02,
       1.15732784e-02, 1.14595050e-02, 1.12290572e-02, 1.10955712e-02,
       1.09763472e-02, 1.09664173e-02, 1.08013630e-02, 1.07163253e-02,
       1.06965233e-02, 1.06243926e-02, 1.05150466e-02, 1.04401201e-02,
       1.04195534e-02, 1.03772631e-02, 1.02585913e-02, 1.02518285e-02,
       1.02343018e-02, 1.02011311e-02, 1.01746044e-02, 1.00893885e-02,
       1.00693090e-02, 1.00007488e-02, 9.97967518e-03, 9.93621541e-03,
       9.91132587e-03, 9.87257873e-03, 9.85864172e-03, 9.85346688e-03,
       9.83507641e-03, 9.82654639e-03, 9.82141035e-03, 9.81950938e-03,
       9.81361594e-03, 9.80760489e-03, 9.80531422e-03, 9.80056163e-03,
       9.79178710e-03, 9.77352236e-03, 9.77198782e-03, 9.75826765e-03,
       9.72396821e-03, 9.71129292e-03, 9.68894621e-03, 9.68294717e-03,
       9.66009263e-03, 9.63768491e-03, 9.62281079e-03, 9.58462673e-03,
       9.56141726e-03, 9.52463771e-03, 9.42925368e-03, 9.41301408e-03,
       9.36144053e-03, 9.30687373e-03, 9.28398704e-03, 9.11455862e-03,
       9.07861857e-03, 9.01571083e-03, 8.94583330e-03, 8.83778340e-03,
       8.72770250e-03, 8.59927945e-03, 8.45704067e-03, 8.40278797e-03,
       8.31065450e-03, 7.87100230e-03, 7.65417979e-03, 7.17339353e-03,
       7.15472018e-03, 6.46609230e-03, 6.11577048e-03, 5.95158857e-03,
       5.31995594e-03, 4.76166961e-03, 4.27250537e-03, 2.37624831e-03,
       1.95586706e-04, 1.09652544e-32, 3.35068907e-33, 1.91665362e-33,
       1.78893920e-33, 1.64352721e-33, 7.42543318e-34, 6.13903002e-34,
       2.44554762e-34, 2.84047797e-35])
```

In [23]:

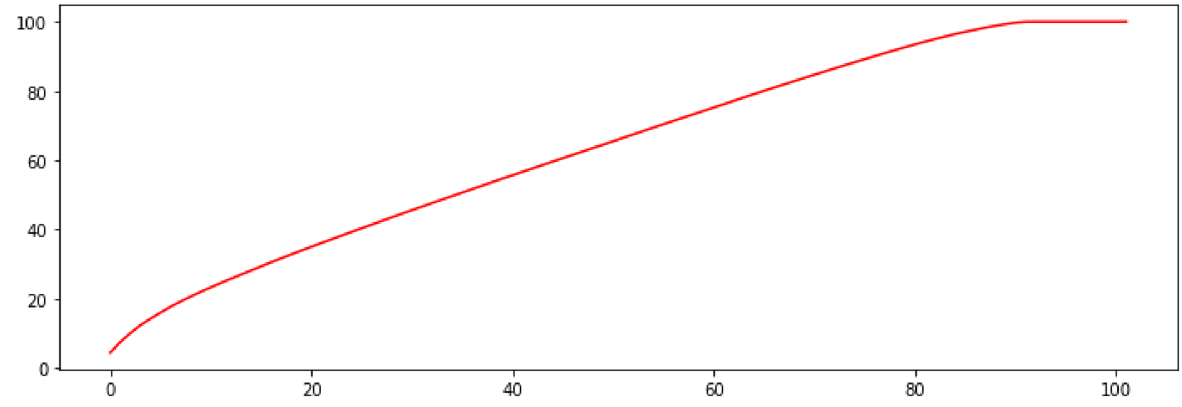
```
# Cumulative variance
var1 = np.cumsum(np.round(var,decimals = 4)*100)
var1
```

Out[23]:

```
array([  4.48,   7.51,  10.08,  12.39,  14.3 ,  16.05,  17.71,  19.23,
        20.64,  22.01,  23.31,  24.58,  25.81,  27.02,  28.21,  29.39,
        30.55,  31.7 ,  32.82,  33.93,  35.03,  36.13,  37.21,  38.28,
        39.35,  40.41,  41.46,  42.5 ,  43.54,  44.58,  45.61,  46.64,
        47.66,  48.68,  49.7 ,  50.71,  51.72,  52.72,  53.72,  54.71,
        55.7 ,  56.69,  57.68,  58.67,  59.65,  60.63,  61.61,  62.59,
        63.57,  64.55,  65.53,  66.51,  67.49,  68.47,  69.45,  70.43,
        71.4 ,  72.37,  73.34,  74.31,  75.28,  76.24,  77.2 ,  78.16,
        79.12,  80.07,  81.01,  81.95,  82.89,  83.82,  84.75,  85.66,
        86.57,  87.47,  88.36,  89.24,  90.11,  90.97,  91.82,  92.66,
        93.49,  94.28,  95.05,  95.77,  96.49,  97.14,  97.75,  98.35,
        98.88,  99.36,  99.79, 100.03, 100.05, 100.05, 100.05, 100.05,
       100.05, 100.05, 100.05, 100.05, 100.05, 100.05])
```

In [24]:

```
# Variance plot for PCA components obtained
plt.figure(figsize=(12,4))
plt.plot(var1,color="red");
```



Let's select 1st 91 columns for model creation, as looking at the data varience we understand that we get 99.36% of the data in 1st 91 columns

In [25]:

```
finaltrain = pd.concat([pd.DataFrame(salary_train_pca_values[:,0:90]),
                     salary_train[['Salary']]], axis = 1)
finaltrain
```

Out[25]:

	0	1	2	3	4	5	6	7	
0	-0.550838	-2.381650	-0.591921	1.433211	0.340516	1.940931	-0.055056	0.515267	-(
1	2.819158	-1.370855	-0.048113	0.060772	0.148609	0.494097	0.971070	0.977166	-1
2	-0.793832	0.871804	-1.202131	-0.513685	0.452004	0.675599	-2.665003	0.041210	(
3	0.732942	2.502494	0.763990	-0.149514	-3.056486	0.166519	0.140931	-2.430582	(
4	-1.070350	-1.638424	4.542395	0.260940	0.650488	-2.473710	2.483233	-2.795740	-(
...	...	...	...	...	...	...	...	...	
30156	-0.766825	-1.596732	-0.037113	-0.576608	0.331904	-3.490439	2.320920	-0.117277	-(
30157	1.704432	1.785328	-0.594925	-0.692498	-0.411458	-1.251061	-0.505186	-0.513443	1
30158	-2.378351	-0.798690	0.339106	-4.067833	0.281604	-1.215646	-0.253538	1.090962	(
30159	-1.975477	1.193052	-1.828994	1.598638	-0.633416	-0.187607	0.789310	0.860548	(
30160	0.762132	-1.772009	0.536972	-2.417460	0.352371	-1.720856	2.466035	1.201089	(

30161 rows × 91 columns

In [26]:

```
finaltest = pd.concat([pd.DataFrame(salary_test_pca_values[:,0:90]),
                       salary_test[['Salary']]], axis = 1)
finaltest
```

Out[26]:

	0	1	2	3	4	5	6	7	
0	-2.242938	2.603181	-0.327617	2.062024	-3.245808	0.812560	-0.442784	-1.685650	C
1	2.226904	1.594715	-0.732083	-0.582928	0.054350	0.429844	-0.039662	1.000275	C
2	2.307044	-1.168832	-0.200521	0.097952	-1.635067	2.571033	1.024094	-0.651146	-1
3	1.080755	1.260533	0.937857	-0.024262	-3.174398	-0.970844	-0.340336	-1.973938	-(
4	-1.759546	1.778058	-1.256430	0.402017	0.646302	1.112461	-0.656860	0.304387	1
...	...	...	...	...	...	...	...	...	
15055	-0.603437	-1.380653	-1.217944	3.023232	0.463631	0.320846	0.117746	-0.907320	1
15056	-1.670443	-3.020764	0.059822	-0.326201	1.744026	-0.876546	-1.136618	-0.761872	C
15057	2.395832	-1.468597	-0.209425	1.122402	0.364606	-0.944486	0.190120	-1.382649	C
15058	-1.140395	-1.036781	2.580795	2.251643	-1.495169	-1.684794	-0.913144	2.031935	-C
15059	3.384451	-1.954816	-0.172792	0.603588	0.377618	0.007115	-0.009551	-0.042785	-1

15060 rows × 91 columns

In [27]:

```
# Since the training dataset is huge, we'll use some part of it for the training purpose, t

# For train dataset
array = finaltrain.values
X = array[0:1000,0:90]
Y = array[0:1000,90]
```

In [28]:

```
# For test dataset
x = finaltest.values[0:1000,0:90]
y = finaltest.values[0:1000,90]
```

Since the training and test datasets are separately given in the problem, we don't need to split the data into train and test here.

SVM Classification

Let's use Grid search CV to find out best value for params

In [29]:

```
clf = SVC()
param_grid = [{'kernel':['rbf'],'gamma':[0.9,0.5,0.1],'C':[1,10,100] },
             {'kernel':['linear'],'C':[1,10,100]}]
gsv = GridSearchCV(clf,param_grid,cv=10,n_jobs=-1)
gsv.fit(X,Y)

gsv.best_params_ , gsv.best_score_
```

Out[29]:

```
({'C': 10, 'kernel': 'linear'}, 0.8220000000000001)
```

In [30]:

```
#SVM Clasification
clf = SVC(C=10, kernel='linear')
clf.fit(x,y)
results = clf.score(x,y)
print(np.round(results, 4))
```

```
0.846
```

The Model accuracy by SVM classification is 85%

In [ ]: