In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split,cross_val_score
```

In [2]:

```
train = pd.read_csv('SalaryData_Train.csv')
test = pd.read_csv('SalaryData_Test.csv')
```

In [3]:

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

30161 rows × 14 columns

In [4]:

```
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 [5]:

```
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]:

```
train.describe()
```

Out[6]:

	age	educationno	capitalgain	capitalloss	hoursperweek
count	30161.000000	30161.000000	30161.000000	30161.000000	30161.000000
mean	38.438115	10.121316	1092.044064	88.302311	40.931269
std	13.134830	2.550037	7406.466611	404.121321	11.980182
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	47.000000	13.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	4356.000000	99.000000

In [7]:

```
test
```

Out[7]:

	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 [8]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15060 entries, 0 to 15059
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            15060 non-null  int64 
 1   workclass      15060 non-null  object
 2   education      15060 non-null  object
 3   educationno    15060 non-null  int64 
 4   maritalstatus  15060 non-null  object
 5   occupation     15060 non-null  object
 6   relationship   15060 non-null  object
 7   race           15060 non-null  object
 8   sex            15060 non-null  object
 9   capitalgain    15060 non-null  int64 
 10  capitalloss    15060 non-null  int64 
 11  hoursperweek   15060 non-null  int64 
 12  native         15060 non-null  object
 13  Salary         15060 non-null  object
dtypes: int64(5), object(9)
memory usage: 1.6+ MB
```

In [9]:

```
test.isna().sum()
```

Out[9]:

```
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 [10]:

```
test.describe()
```

Out[10]:

	age	educationno	capitalgain	capitalloss	hoursperweek
count	15060.000000	15060.000000	15060.000000	15060.000000	15060.000000
mean	38.768327	10.112749	1120.301594	89.041899	40.951594
std	13.380676	2.558727	7703.181842	406.283245	12.062831
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	48.000000	13.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	3770.000000	99.000000

In [11]:

```
# frequency for categorical fields
```

In [13]:

```
category_col =['workclass', 'education','maritalstatus', 'occupation', 'relationship', 'rac
for c in category_col:
    print (c)
    print (train[c].value_counts())
    print('\n')
```

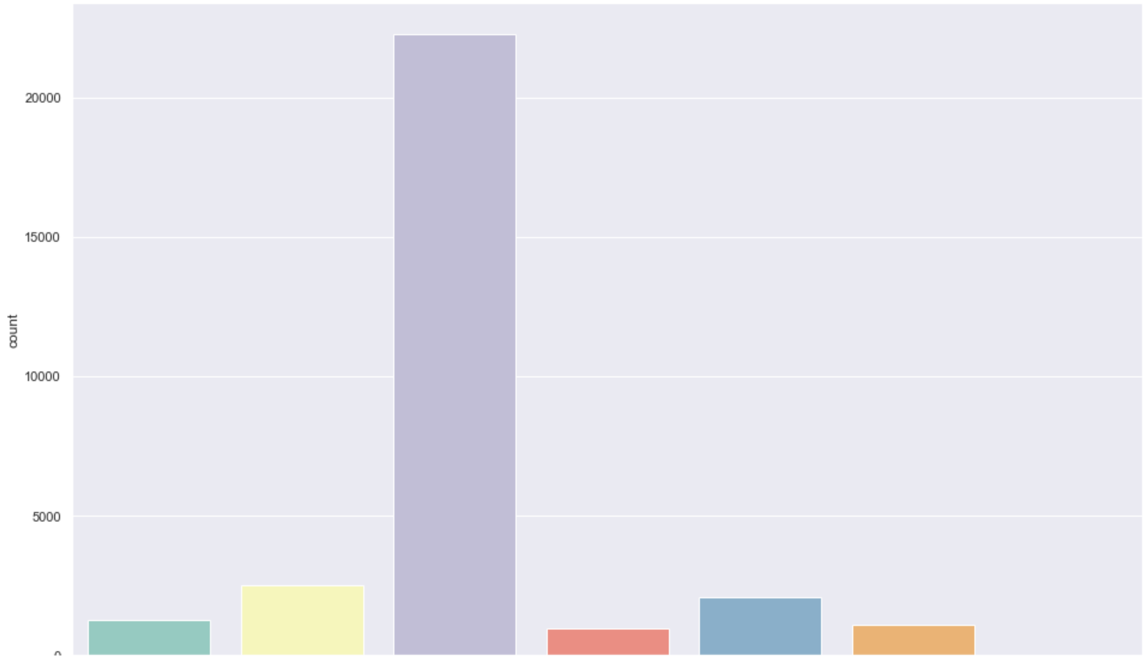
```
workclass
 Private             22285
 Self-emp-not-inc     2499
 Local-gov            2067
 State-gov            1279
 Self-emp-inc         1074
 Federal-gov           943
 Without-pay            14
Name: workclass, dtype: int64


education
 HS-grad         9840
 Some-college    6677
 Bachelors       5044
 Masters         1627
 Assoc-voc       1307
 11th            1048
 Assoc-acdm      1008
```

In [14]:

```
#Countplot for all categorical columns
```

In [16]:

```
sns.set(rc={'figure.figsize':(16,10)})
cat_col = ['workclass', 'education','maritalstatus', 'occupation', 'relationship', 'race',
for col in cat_col:
    plt.figure() # this create a new figure on which our plot will appear
    sns.countplot(x=col, data=train,palette='Set3');
    plt.show()
```



In [17]:

```
train[['Salary','age']].groupby(['Salary'],as_index=False).mean().sort_values(by='age', asc
```

Out[17]:

	Salary	age
1	>50K	43.959110
0	<=50K	36.608264

In [18]:

```
##Feature Encoding
```

In [19]:

```
from sklearn.preprocessing import LabelEncoder
```

In [20]:

```
train = train.apply(LabelEncoder().fit_transform)
train.head()
```

Out[20]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	22	5	9	12	4	0	1	4	1
1	33	4	9	12	2	3	0	4	1
2	21	2	11	8	0	5	1	4	1
3	36	2	1	6	2	5	0	2	1
4	11	2	9	12	2	9	5	2	0

In [21]:

```
test = test.apply(LabelEncoder().fit_transform)
test.head()
```

Out[21]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	8	2	1	6	4	6	3	2	1
1	21	2	11	8	2	4	0	4	1
2	11	1	7	11	2	10	0	4	1
3	27	2	15	9	2	6	0	2	1
4	17	2	0	5	4	7	1	4	1

Multinomial naive bays model on SalaryDataTrain

In [22]:

```
#Train test split
```

In [24]:

```
drop_elements = ['education','native','Salary']
x = train.drop(drop_elements,axis=1)
y = train['Salary']
```

In [26]:

```
x
```

Out[26]:

	age	workclass	educationno	maritalstatus	occupation	relationship	race	sex	capitalg
0	22	5	12	4	0	1	4	1	
1	33	4	12	2	3	0	4	1	
2	21	2	8	0	5	1	4	1	
3	36	2	6	2	5	0	2	1	
4	11	2	12	2	9	5	2	0	
...	...	...	...	...	...	...	...	...	
30156	10	2	11	2	12	5	4	0	
30157	23	2	8	2	6	0	4	1	
30158	41	2	8	6	0	4	4	0	
30159	5	2	8	4	0	3	4	1	
30160	35	3	8	2	3	5	4	0	

30161 rows × 11 columns

In [27]:

```
y
```

Out[27]:

```
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 [28]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33, random_state=42)
```

Building Modl

In [29]:

```
#Preparing a naive bays model on training dataset
```

In [30]:

```
from sklearn.naive_bayes import MultinomialNB as MB
from sklearn.naive_bayes import GaussianNB as GB

#Muktinomial naive bayes
classifier_mb = MB()
classifier_mb.fit(x_train,y_train)
```

Out[30]:

```
MultinomialNB()
```

In [31]:

```
score_multinomial = classifier_mb.score(x_test,y_test)
print('The accuracy of Multinomial Naive Bayes is', score_multinomial)
```

```
The accuracy of Multinomial Naive Bayes is 0.7796865581675708
```

Building Gauessian model

In [32]:

```
classifier_gb = GB()
classifier_gb.fit(x_train,y_train)
```

Out[32]:

```
GaussianNB()
```

In [33]:

```
score_gaussian =  classifier_gb.score(x_test,y_test)
print('The accuracy of Gaussian Naive Bayes is', score_gaussian)
```

```
The accuracy of Gaussian Naive Bayes is 0.812035362668274
```

In [37]:

```
## Testing Multinomial Naive Bays model on SalaryDataTest
```

In [35]:

```
drop_elements = ['education', 'native', 'Salary']
X = test.drop(drop_elements, axis=1)

Y = test['Salary']
```

In [38]:

```
##Testing Gaussian Naive Bays model on SalaryDataTest
```

In [36]:

```
from sklearn import metrics

# make predictions
new_prediction = classifier_gb.predict(X)
# summarize the fit of the model
print(metrics.classification_report(Y, new_prediction))
print(metrics.confusion_matrix(Y, new_prediction))

print("Accuracy:",metrics.accuracy_score(Y, new_prediction))
print("Precision:",metrics.precision_score(Y, new_prediction))
print("Recall:",metrics.recall_score(Y, new_prediction))
```

```
              precision    recall  f1-score   support

           0       0.84      0.93      0.88     11360
           1       0.69      0.45      0.54      3700

    accuracy                           0.81     15060
   macro avg       0.76      0.69      0.71     15060
weighted avg       0.80      0.81      0.80     15060

[[10604   756]
 [ 2038  1662]]
Accuracy: 0.8144754316069057
Precision: 0.6873449131513648
Recall: 0.4491891891891892
```

In [ ]:

```

```