Assignment-12-Naive-Bayes

In [1]:

```
import pandas as pd
import numpy as np
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]:

```
# Import dataset
df_train = pd.read_csv('C:/Users/LENOVO/Documents/Custom Office Templates/SalaryData_Train.
df_train.head()
```

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	Ma
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Ma
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Ma
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Ma
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fema

In [3]:

```
df_test = pd.read_csv('C:/Users/LENOVO/Documents/Custom Office Templates/SalaryData_Test.cs
df_test.head()
```

Out[3]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White	Male

EDA & Data Preprocessing

In [4]:

```
#Check datatypes
df_train.dtypes
```

Out[4]:

```
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 [5]:

```
df_train.shape
```

Out[5]:

```
(30161, 14)
```

In [6]:

```
df_train.isnull().sum()
```

Out[6]:

```
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
```

No null values in salary_train dataset

In [7]:

```
df_train.value_counts('Salary')
```

Out[7]:

```
Salary
 <=50K    22653
 >50K      7508
dtype: int64
```

In [8]:

```
# frequency for categorical fields
category_col =['workclass', 'education','maritalstatus', 'occupation', 'relationship', 'rac
for c in category_col:
    print (c)
    print (df_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
```

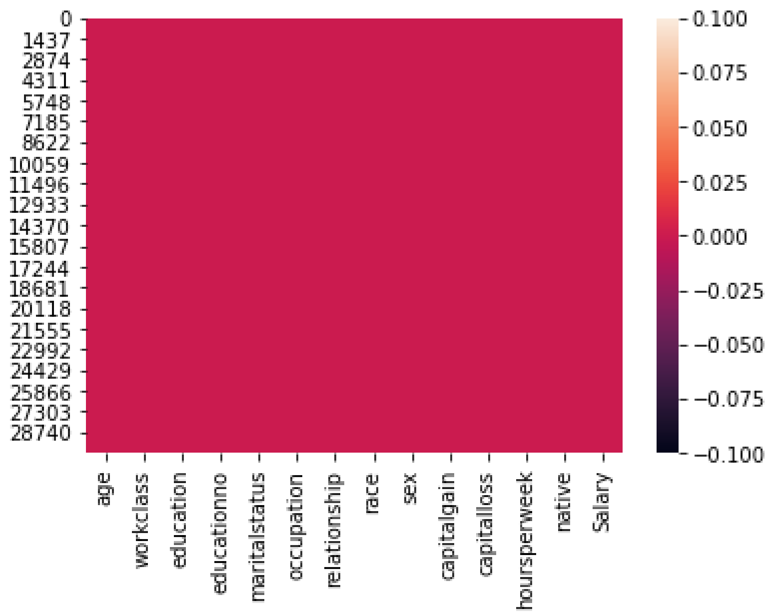
Visualization

In [9]:

```
import seaborn as sns
sns.heatmap(df_train.isnull())
```

Out[9]:

<AxesSubplot:>



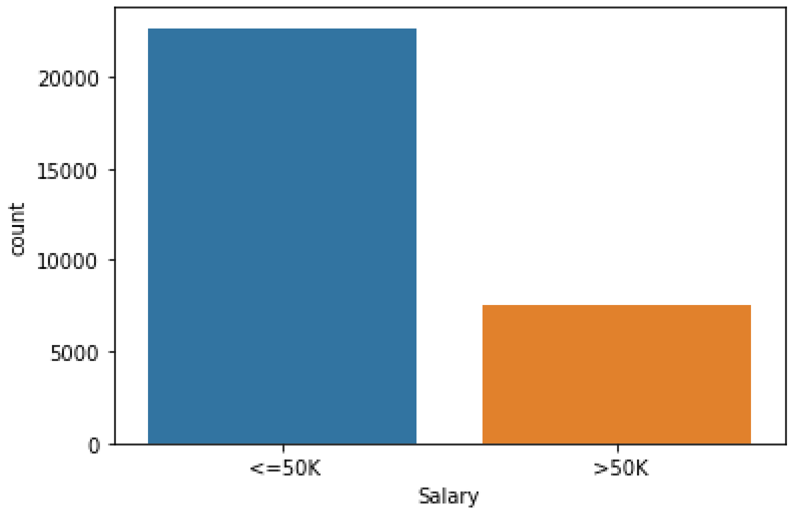
In [10]:

```
sns.countplot(df_train["Salary"])
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: Pass the following variable as a keyword arg: x. From version 0.1
2, the only valid positional argument will be `data`, and passing other argu
ments without an explicit keyword will result in an error or misinterpretati
on.
  warnings.warn(
```

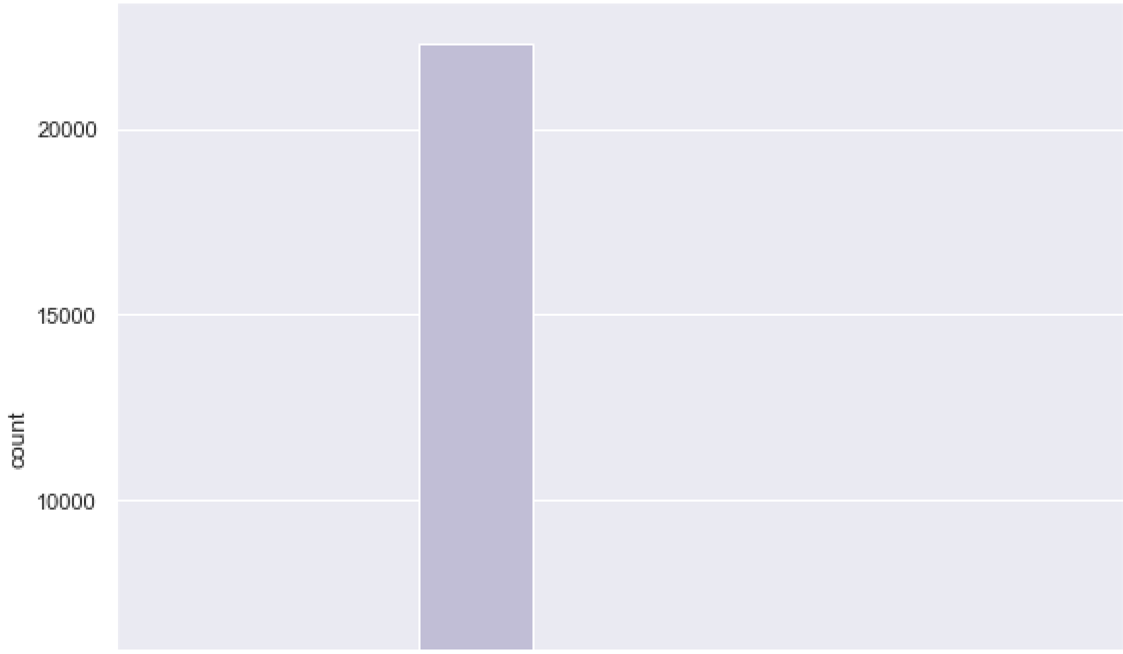
Out[10]:

```
<AxesSubplot:xlabel='Salary', ylabel='count'>
```



In [11]:

```
# countplot for all categorical columns
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(rc={'figure.figsize':(9,8)})
cat_col = ['workclass', 'education','maritalstatus', 'occupation', 'relationship', 'race',
for col in cat_col:
    plt.figure() #this creates a new figure on which your plot will appear
    sns.countplot(x = col, data = df_train, palette = 'Set3');
```



Printing unique values from each categorical columns

In [12]:

```
print('workclass',df_train.workclass.unique())
print('education',df_train.education.unique())
print('maritalstatus',df_train['maritalstatus'].unique())
print('occupation',df_train.occupation.unique())
print('relationship',df_train.relationship.unique())
print('race',df_train.race.unique())
print('sex',df_train.sex.unique())
print('native',df_train['native'].unique())
print('Salary',df_train.Salary.unique())
```

```
workclass [' State-gov' ' Self-emp-not-inc' ' Private' ' Federal-gov' ' Loca
l-gov'
 ' Self-emp-inc' ' Without-pay']
education [' Bachelors' ' HS-grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' 7th-8th' ' Doctorate' ' Assoc-voc' ' Prof-school'
 ' 5th-6th' ' 10th' ' Preschool' ' 12th' ' 1st-4th']
maritalstatus [' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']
occupation [' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-s
pecialty'
 ' Other-service' ' Sales' ' Transport-moving' ' Farming-fishing'
 ' Machine-op-inspct' ' Tech-support' ' Craft-repair' ' Protective-serv'
 ' Armed-Forces' ' Priv-house-serv']
relationship [' Not-in-family' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']
race [' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo' ' Othe
r']
sex [' Male' ' Female']
native [' United-States' ' Cuba' ' Jamaica' ' India' ' Mexico' ' Puerto-Ric
o'
 ' Honduras' ' England' ' Canada' ' Germany' ' Iran' ' Philippines'
 ' Poland' ' Columbia' ' Cambodia' ' Thailand' ' Ecuador' ' Laos'
 ' Taiwan' ' Haiti' ' Portugal' ' Dominican-Republic' ' El-Salvador'
 ' France' ' Guatemala' ' Italy' ' China' ' South' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary']
Salary [' <=50K' ' >50K']
```

In [13]:

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

Out[13]:

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

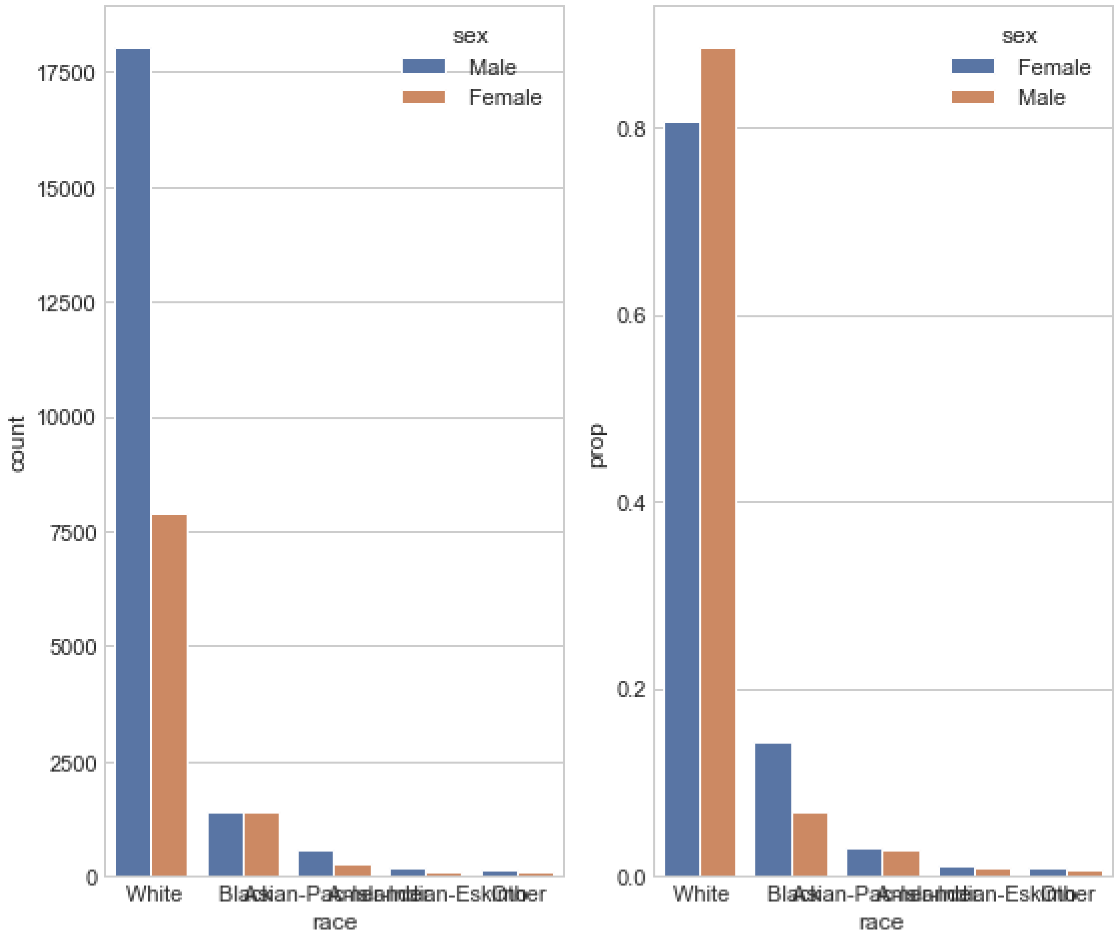
In [14]:

```
plt.style.use('seaborn-whitegrid')
x, y, hue = "race", "prop", "sex"
#hue_order = ["Male", "Female"]
plt.figure(figsize=(20,10))
f, axes = plt.subplots(1, 2)
sns.countplot(x=x, hue=hue, data=df_train, ax=axes[0])

prop_df = (df_train[x]
           .groupby(df_train[hue])
           .value_counts(normalize=True)
           .rename(y)
           .reset_index())

sns.barplot(x=x, y=y, hue=hue, data=prop_df, ax=axes[1])
```

Out[14]:

```
<AxesSubplot:xlabel='race', ylabel='prop'>
```

```
<Figure size 1440x720 with 0 Axes>
```



Feature encoding

In [15]:

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

In [16]:

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

Out[16]:

	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 [17]:

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

Out[17]:

	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

Test-Train-Split

In [18]:

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

In [19]:

```
y = df_train['Salary']
```

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Building Multinomial Naive Bays Model

In [21]:

```
# Preparing a naive bayes model on training data set

from sklearn.naive_bayes import MultinomialNB as MB

# Multinomial Naive Bayes
classifier_mb = MB()
classifier_mb.fit(X_train, y_train)
```

Out[21]:

```
MultinomialNB()
```

In [22]:

```
score_multinomial_train = classifier_mb.score(X_train,y_train)
print('The accuracy of Gaussian Naive Bayes is', score_multinomial_train)
```

```
The accuracy of Gaussian Naive Bayes is 0.7788390161825111
```

In [23]:

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

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

Testing Multinomial Naive Bayes model on SalaryData_Test.csv

In [24]:

```
from sklearn import metrics

drop_elements = ['education', 'native', 'Salary']
X_new = df_test.drop(drop_elements, axis=1)

y_new = df_test['Salary']
```

In [25]:

```
# make predictions
new_prediction = classifier_mb.predict(X_new)
# summarize the fit of the model
print(metrics.classification_report(y_new, new_prediction))
print(metrics.confusion_matrix(y_new, new_prediction))

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

```
              precision    recall  f1-score   support

           0       0.80      0.94      0.87     11360
           1       0.61      0.30      0.40      3700

    accuracy                           0.78     15060
   macro avg       0.71      0.62      0.63     15060
weighted avg       0.76      0.78      0.75     15060

[[10648   712]
 [ 2587  1113]]
Accuracy: 0.7809428950863214
Precision: 0.6098630136986302
Recall: 0.3008108108108108
```

Building Gaussian Naive Bayes Model

In [26]:

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB as GB

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

Out[26]:

```
GaussianNB()
```

In [27]:

```
score_gaussian_train = classifier_gb.score(X_train,y_train)
print('The accuracy of Gaussian Naive Bayes is', score_gaussian_train)
```

```
The accuracy of Gaussian Naive Bayes is 0.8108576235957836
```

In [28]:

```
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
```

Testing Gaussian Naive Bays model on SalaryData_Test.csv

In [29]:

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

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

Compare train and test accuracy

The training-set accuracy score is 0.8108 while the test-set accuracy to be 0.8120.

These two values are quite comparable. So, there is no sign of overfitting

k-Fold Cross Validation

In [31]:

```
# Applying 10-Fold Cross Validation

from sklearn.model_selection import cross_val_score

scores = cross_val_score(classifier_gb, X_train, y_train, cv = 10, scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))
```

```
Cross-validation scores:[0.80554181 0.81840673 0.80999505 0.81296388 0.81642
751 0.80356259
 0.79020287 0.82178218 0.80643564 0.81683168]
```

In [32]:

```
# compute Average cross-validation score

print('Average cross-validation score: {:.4f}'.format(scores.mean()))
```

```
Average cross-validation score: 0.8102
```

In [ ]:

```

```