In [46]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

import warnings
warnings.filterwarnings('ignore')
```

In [47]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

In [176]:

```
# common function ,chi2 and Anova
from sklearn.feature_selection import SelectKBest,chi2,f_regression
# Principal component analysis
from sklearn.decomposition import PCA
```

In [6]:

```
df = pd.read_csv("../../CSV/salary.csv",index_col=0)
```

In [7]:

```
df.head()
```

Out[7]:

	age	Workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fe

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
age               0
Workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Income            0
dtype: int64
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1032 entries, 0 to 1119
Data columns (total 15 columns):
age               1032 non-null int64
Workclass         1032 non-null object
fnlwgt            1032 non-null int64
education         1032 non-null object
education-num     1032 non-null int64
marital-status    1032 non-null object
occupation        1032 non-null object
relationship      1032 non-null object
race              1032 non-null object
sex               1032 non-null object
capital-gain      1032 non-null int64
capital-loss      1032 non-null int64
hours-per-week    1032 non-null int64
native-country    1032 non-null object
Income            1032 non-null int64
dtypes: int64(7), object(8)
memory usage: 129.0+ KB
```

In [10]:

```
df.describe(include="all")
```

Out[10]:

	age	Workclass	fnlwgt	education	education-num	marital-status	occupation	re
count	1032.000000	1032	1.032000e+03	1032	1032.000000	1032	1032	
unique	NaN	6	NaN	16	NaN	7	14	
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty	
freq	NaN	766	NaN	338	NaN	472	143	
mean	37.954457	NaN	1.918318e+05	NaN	10.221899	NaN	NaN	
std	12.825353	NaN	1.074243e+05	NaN	2.501636	NaN	NaN	
min	17.000000	NaN	2.117400e+04	NaN	1.000000	NaN	NaN	
25%	28.000000	NaN	1.155792e+05	NaN	9.000000	NaN	NaN	
50%	37.000000	NaN	1.807065e+05	NaN	10.000000	NaN	NaN	
75%	46.000000	NaN	2.461932e+05	NaN	13.000000	NaN	NaN	
max	90.000000	NaN	1.033222e+06	NaN	16.000000	NaN	NaN	

In [11]:

```
df["Income"].value_counts()
```

Out[11]:

```
0    778
1    254
Name: Income, dtype: int64
```

Label Encoding

In [12]:

```
cat_df = df.select_dtypes("object")
num_df = df.select_dtypes("int64")
```

In [14]:

```
for col in cat_df:
    le = LabelEncoder()
    cat_df[col] = le.fit_transform(cat_df[col])
```

In [16]:

```
cat_df.head()
```

Out[16]:

	Workclass	education	marital-status	occupation	relationship	race	sex	native-country
0	5	9	4	0	1	4	1	28
1	4	9	2	3	0	4	1	28
2	2	11	0	5	1	4	1	28
3	2	1	2	5	0	2	1	28
4	2	9	2	9	5	2	0	4

In [17]:

```
df = pd.concat([cat_df,num_df],axis=1)
```

In [18]:

```
df.head()
```

Out[18]:

	Workclass	education	marital-status	occupation	relationship	race	sex	native-country	age	fnlwgt	e
0	5	9	4	0	1	4	1	28	39	77516	
1	4	9	2	3	0	4	1	28	50	83311	
2	2	11	0	5	1	4	1	28	38	215646	
3	2	1	2	5	0	2	1	28	53	234721	
4	2	9	2	9	5	2	0	4	28	338409	

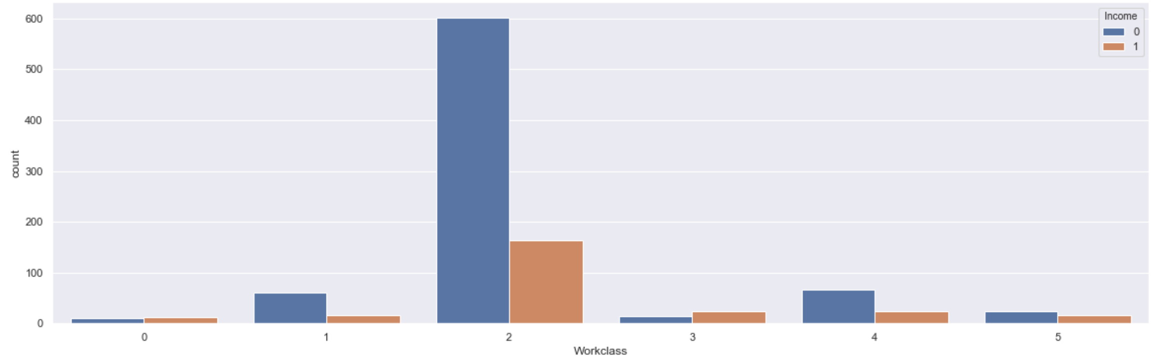
EDA

In [19]:

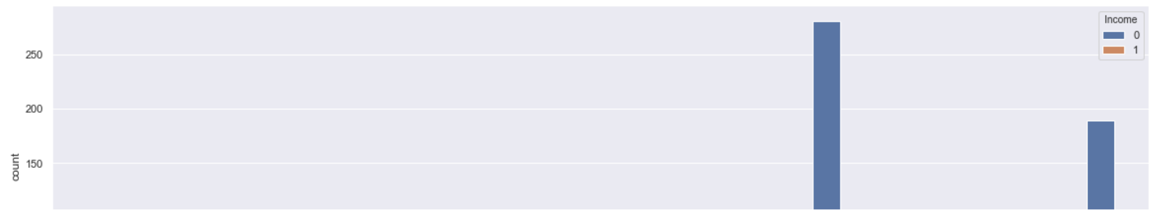
```
cat_col = ("Workclass,education,marital-status,occupation,relationship,race,sex,native-coun
```

In [20]:

```
for col in cat_col:
    plt.figure(figsize=(20,6))
    sns.countplot(data=df,x=col,hue="Income")
    plt.show()
    print("-----------------------------------------------------------------------")
```




```
-----------------------------------------------------------------------
```

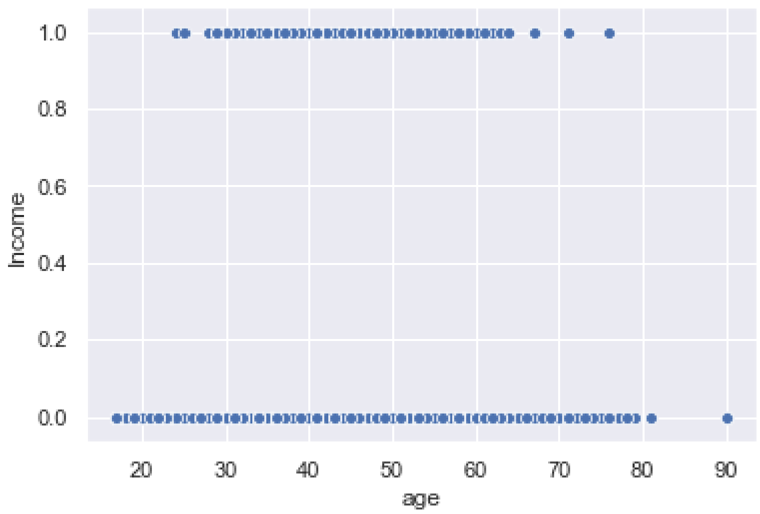


In [21]:

```
num_col = ("age,fnlwgt,education-num,capital-gain,capital-loss,hours-per-week").split(",")
```

In [22]:

```
for col in num_col:
    plt.figure()
    sns.scatterplot(data=df,x=col,y="Income")
    plt.show()
    print("-------------------------------------------------------------------------------"
```




```
-------------------------------------------------------------------------
-----
```

Baseline model

In [94]:

```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [95]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

In [96]:

```
def train_model(X_train,X_test):
    log = LogisticRegression()
    log.fit(X_train,y_train)
    y_pred = log.predict(X_test)
    print(classification_report(y_test,y_pred))
```

In [97]:

```
train_model(X_train,X_test)
```

```
              precision    recall  f1-score   support

           0       0.81      0.98      0.89       238
           1       0.77      0.24      0.36        72

    accuracy                           0.81       310
   macro avg       0.79      0.61      0.62       310
weighted avg       0.80      0.81      0.76       310
```

Filter Method

In [186]:

```
def create_model(model):
    X_train_model = model.fit_transform(X_train,y_train)
    X_test_model = model.transform(X_test)
    train_model(X_train_model,X_test_model)
    return model
```

Chi2 Test

In [177]:

```
# chi2
#from sklearn.feature_selection import chi2
chi2 = SelectKBest(score_func=chi2, k=11)
```

In [178]:

```
create_model(chi2)
```

```
              precision    recall  f1-score   support

           0       0.80      0.96      0.88       238
           1       0.64      0.22      0.33        72

    accuracy                           0.79       310
   macro avg       0.72      0.59      0.60       310
weighted avg       0.77      0.79      0.75       310
```

In [163]:

```
chi2.scores_
```

Out[163]:

```
array([1.60667219e+00, 6.46272080e-01, 2.85524856e+01, 2.69228910e-01,
       4.28508247e+01, 6.51262333e-01, 3.88273433e+00, 5.29629278e-02,
       1.43215169e+02, 2.08905154e+04, 4.51886417e+01, 7.86853758e+05,
       1.98133155e+04, 1.23675492e+02])
```

In [35]:

```
X.columns
```

Out[35]:

```
Index(['Workclass', 'education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'native-country', 'age', 'fnlwgt',
       'education-num', 'capital-gain', 'capital-loss', 'hours-per-week'],
      dtype='object')
```

In [230]:

```
chi2.get_support()
```

Out[230]:

```
array([ True, False,  True, False,  True,  True,  True, False,  True,
        True,  True,  True,  True,  True])
```

In [ ]:

```
# Less imp features - education,occupation,native_country
```

Anova Test

In [188]:

```
anova = SelectKBest(score_func=f_regression,k=11)
```

In [189]:

```
anova = create_model(anova)
```

```
              precision    recall  f1-score   support

           0       0.84      0.96      0.90       238
           1       0.74      0.40      0.52        72

    accuracy                           0.83       310
   macro avg       0.79      0.68      0.71       310
weighted avg       0.82      0.83      0.81       310
```

In [191]:

```
anova.scores_
```

Out[191]:

```
array([ 3.855057  ,  0.51247514, 34.01671063,  0.10304801, 24.01569722,
        3.22653221, 12.36335806,  0.08849767, 35.87430298,  0.34703723,
       87.75938546, 86.66744615, 11.18286257, 40.91992553])
```

In [192]:

```
X.columns
```

Out[192]:

```
Index(['Workclass', 'education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'native-country', 'age', 'fnlwgt',
       'education-num', 'capital-gain', 'capital-loss', 'hours-per-week'],
      dtype='object')
```

In [193]:

```
anova.get_support()
```

Out[193]:

```
array([ True,  True,  True, False,  True,  True,  True, False,  True,
       False,  True,  True,  True,  True])
```

In [ ]:

```
# Less imp features - occupation,native-country,fnlwgt
```

Wrapper Method

In [194]:

```
features = df.columns.tolist()[:-1]
```

In [198]:

```
features
```

Out[198]:

```
['Workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native-country',
 'age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week']
```

Forward Selection

In [200]:

```
cols = []
i = len(cols) + 1
for col in features:
    cols.append(col)

    X = df[cols]
    y = df["Income"]

    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)

    print("Iteration :",i, "  Columns :",cols)
    train_model(X_train,X_test)
    i += 1
    print("--------------------------------------------------------------------------------
```

```
Iteration : 1   Columns : ['Workclass']
              precision    recall  f1-score   support

           0       0.77      1.00      0.87       238
           1       0.00      0.00      0.00        72

    accuracy                           0.77       310
   macro avg       0.38      0.50      0.43       310
weighted avg       0.59      0.77      0.67       310

--------------------------------------------------------------------------
--------
Iteration : 2   Columns : ['Workclass', 'education']
              precision    recall  f1-score   support

           0       0.77      1.00      0.87       238
           1       0.00      0.00      0.00        72

    accuracy                           0.77       310
```

Backward Selection

In [202]:

```
cols = []
cols.extend(features)
i = len(cols)
for col in features:

    X = df[cols]
    y = df["Income"]

    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)

    print("Iteration :",i, "  Columns :",cols)
    train_model(X_train,X_test)
    i -= 1
    cols.remove(col)
    print("--------------------------------------------------------------------------------
```

```
Iteration : 14   Columns : ['Workclass', 'education', 'marital-status', 'o
ccupation', 'relationship', 'race', 'sex', 'native-country', 'age', 'fnlwg
t', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
              precision    recall  f1-score   support

           0       0.81      0.98      0.89       238
           1       0.77      0.24      0.36        72

    accuracy                           0.81       310
   macro avg       0.79      0.61      0.62       310
weighted avg       0.80      0.81      0.76       310

--------------------------------------------------------------------------
--------
Iteration : 13   Columns : ['education', 'marital-status', 'occupation',
'relationship', 'race', 'sex', 'native-country', 'age', 'fnlwgt', 'educati
on-num', 'capital-gain', 'capital-loss', 'hours-per-week']
              precision    recall  f1-score   support
```

Pricipal Component Analysis (PCA)

In [203]:

```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
```

In [222]:

```
pca = PCA(n_components=10,random_state=1)
```

In [224]:

```
pca = create_model(pca)
```

```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89       238
           1       0.71      0.40      0.51        72

    accuracy                           0.82       310
   macro avg       0.77      0.68      0.70       310
weighted avg       0.81      0.82      0.80       310
```

In [225]:

```
pca.noise_variance_
```

Out[225]:

```
0.9260945320129395
```

In [227]:

```
pc = pca.components_
```

In [228]:

```
pc[0]
```

Out[228]:

```
array([ 3.40042108e-07, -1.33962105e-06,  6.89948962e-07, -2.67201320e-07,
        9.49268155e-07, -7.71548627e-07, -6.75004521e-08, -2.64531132e-06,
       -7.24832430e-06,  9.99999750e-01, -1.41921124e-06,  7.07377252e-04,
       -1.95570803e-05, -8.45583433e-06])
```

In [229]:

```
X.columns
```

Out[229]:

```
Index(['Workclass', 'education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'native-country', 'age', 'fnlwgt',
       'education-num', 'capital-gain', 'capital-loss', 'hours-per-week'],
      dtype='object')
```

In [ ]:

```
# Final conclusion: occupation,native-country,fnlwgt are less important features may be rem
# however there is domain knowledge also matters for more preprocessing of data.
```