In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder,MinMaxScaler,StandardScaler
```

In [5]:

```
df=pd.read_csv(r"C:\Users\dtdee\OneDrive\Desktop\Letsupgrade_Python\Machine_Learning\SVM
```

In [6]:

```
df.head(3)
```

Out[6]:

	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

In [7]:

```
df.shape
```

Out[7]:

```
(32561, 15)
```

In [8]:

```
df.drop(columns=' fnlwgt',axis=1,inplace=True)
```

In [9]:

```
df.head()
```

Out[9]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
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

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              32561 non-null  int64 
 1    workclass       32561 non-null  object
 2    education       32561 non-null  object
 3    education-num   32561 non-null  int64 
 4    marital-status  32561 non-null  object
 5    occupation      32561 non-null  object
 6    relationship    32561 non-null  object
 7    race            32561 non-null  object
 8    sex             32561 non-null  object
 9    capital-gain    32561 non-null  int64 
 10   capital-loss    32561 non-null  int64 
 11   hours-per-week  32561 non-null  int64 
 12   native-country  32561 non-null  object
 13   income          32561 non-null  object
dtypes: int64(5), object(9)
memory usage: 3.5+ MB
```

In [11]:

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

Out[11]:

```
age                0
 workclass         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 [12]:

```
# Checking for duplicate rows or entries

df.duplicated().sum()
```

Out[12]:

```
3465
```

In [13]:

```
#Dropping all the duplicate rows

df.drop_duplicates(inplace=True)
```

In [14]:

```
df.shape
```

Out[14]:

```
(29096, 14)
```

In [15]:

```
df.columns
```

Out[15]:

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

In [16]:

```
# Fixing the column names and there is a space in front of every column name

columns=['age', 'workclass', 'education', 'education-num', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country', 'income']
```

In [17]:

```
df.columns=columns
```

In [18]:

```
df.columns
```

Out[18]:

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

In [19]:

```
df.head(2)
```

Out[19]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	ca
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	

In [20]:

```
for i in df.columns:
    if i in ['workclass', 'education','marital-status','occupation', 'relationship', 'ra
        print(f'\n\nUnique values in the column {i} is :\n',df[i].unique())
```

```


Unique values in the column workclass is :
 [' State-gov' ' Self-emp-not-inc' ' Private' ' Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']


Unique values in the column education is :
 [' Bachelors' ' HS-grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th']


Unique values in the column marital-status is :
 [' Never-married' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']


Unique values in the column occupation is :
 [' Adm-clerical' ' Exec-managerial' ' Handlers-cleaners' ' Prof-specialt
y'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' ' ?'
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv']


Unique values in the column relationship is :
 [' Not-in-family' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']


Unique values in the column race is :
 [' White' ' Black' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo' ' Other']


Unique values in the column sex is :
 [' Male' ' Female']


Unique values in the column native-country is :
 [' United-States' ' Cuba' ' Jamaica' ' India' ' ?' ' Mexico' ' South'
 ' Puerto-Rico' ' Honduras' ' England' ' Canada' ' Germany' ' Iran'
 ' Philippines' ' Italy' ' Poland' ' Columbia' ' Cambodia' ' Thailand'
 ' Ecuador' ' Laos' ' Taiwan' ' Haiti' ' Portugal' ' Dominican-Republic'
 ' El-Salvador' ' France' ' Guatemala' ' China' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary'
 ' Holand-Netherlands']
```

In [ ]:

```

```

In [21]:

```
for i in df.columns:
    if i in ['workclass', 'education','marital-status','occupation', 'relationship', 'ra
            df[i]=df[i].str.strip()
```

In [22]:

```
df.workclass.unique()
```

Out[22]:

```
array(['State-gov', 'Self-emp-not-inc', 'Private', 'Federal-gov',
       'Local-gov', '?', 'Self-emp-inc', 'Without-pay', 'Never-worked'],
      dtype=object)
```

In [23]:

```
df['workclass'].replace({'?':'Without-pay'})
```

Out[23]:

```
0               State-gov
1        Self-emp-not-inc
2                 Private
3                 Private
4                 Private
               ...
32554             Private
32555             Private
32556             Private
32558             Private
32560        Self-emp-inc
Name: workclass, Length: 29096, dtype: object
```

In [24]:

```
df['native-country'].value_counts()
```

Out[24]:

```
United-States                 25721
Mexico                          633
?                               580
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Japan                            62
Guatemala                        62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
Greece                           29
Ecuador                          28
Ireland                          23
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Honduras                         13
Hungary                          13
Scotland                         12
Holand-Netherlands                1
Name: native-country, dtype: int64
```

In [25]:

```
df['occupation'].replace({'?':'others'},inplace=True)
df['native-country'].replace({'?':'some-country'},inplace=True)
```

In [26]:

```
x=df['income'].value_counts()
```

In [27]:

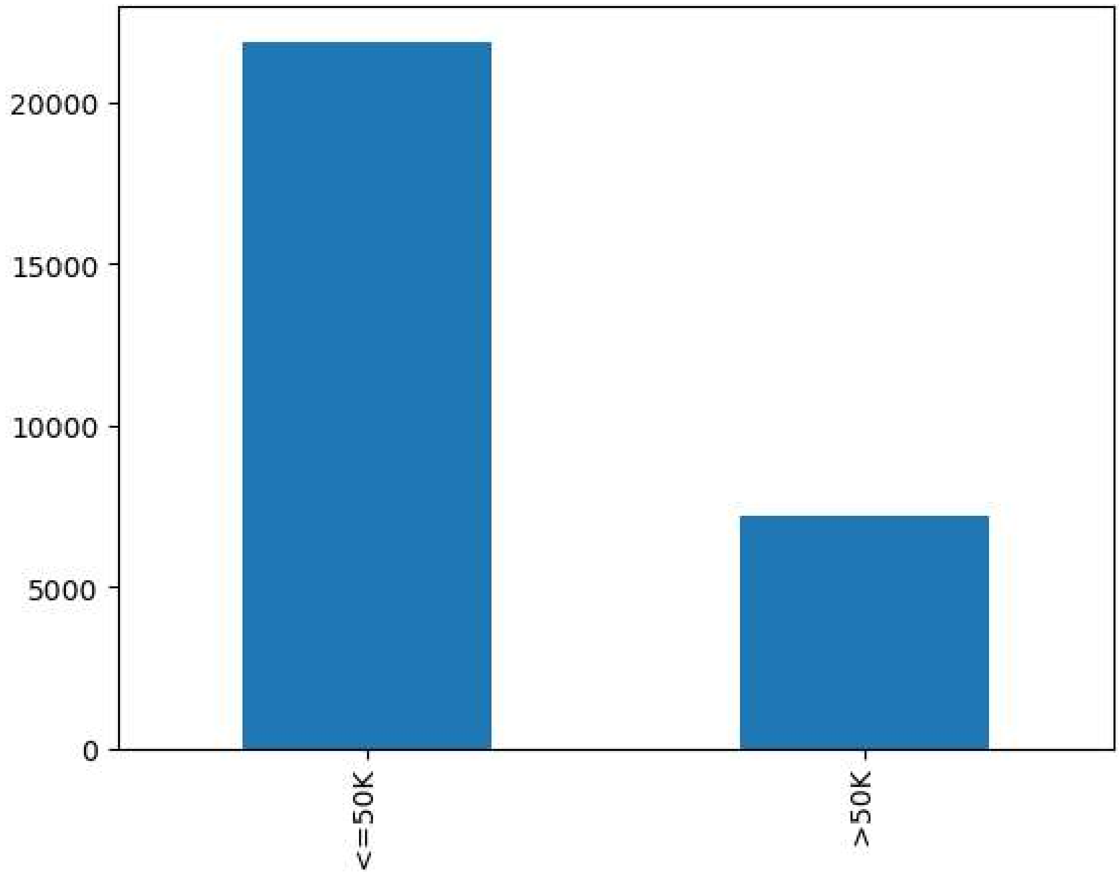
```
df['income'].replace({'<=50K:0':0,'>50K':1},inplace=True)
```

In [28]:

```
x.plot(kind='bar')
```

Out[28]:

```
<AxesSubplot:>
```



In [29]:

```
df.head(2)
```

Out[29]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	ca
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	

In [30]:

```
bin=[16,25,50,100]
labels=['young','adult','old']

df['age_type']=pd.cut(df['age'],bins=bin,labels=labels)
```

In [31]:

```
df.columns
```

Out[31]:

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

In [32]:

```
df=df[[ 'workclass', 'education', 'education-num', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country', 'income',
       'age_type']]
```

In [33]:

```
df.head()
```

Out[33]:

	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capita ga
0	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	21
1	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	

In [34]:

```
df.describe().T
```

Out[34]:

	count	mean	std	min	25%	50%	75%	max
education-num	29096.0	10.102695	2.645194	1.0	9.0	10.0	13.0	16.0
capital-gain	29096.0	1197.802206	7778.225220	0.0	0.0	0.0	0.0	99999.0
capital-loss	29096.0	97.175179	424.008232	0.0	0.0	0.0	0.0	4356.0
hours-per-week	29096.0	40.637820	12.735418	1.0	40.0	40.0	45.0	99.0

In [35]:

```
df.corr()
```

Out[35]:

	education-num	capital-gain	capital-loss	hours-per-week
education-num	1.000000	0.124182	0.080259	0.141446
capital-gain	0.124182	1.000000	-0.035294	0.077704
capital-loss	0.080259	-0.035294	1.000000	0.051636
hours-per-week	0.141446	0.077704	0.051636	1.000000

In [36]:

```
df.income.unique()
```

Out[36]:

```
array([' <=50K', ' >50K'], dtype=object)
```

In [37]:

```
df['income_new']=np.where(df['income']==' <=50K',0,1).astype('int16')
```

In [38]:

```
df.columns
```

Out[38]:

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

In [39]:

```
df=df[['workclass', 'education', 'education-num', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country',
       'age_type', 'income_new']]
df
```

Out[39]:

	workclass	education	education-num	marital-status	occupation	relationship	race	sex
0	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
1	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male
4	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female
...	...	...	...	...	...	...	...	...
32554	Private	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male
32555	Private	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Male
32556	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female
32558	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female
32560	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female

29096 rows × 14 columns

In [40]:

```
for i in df.columns:
    if i in ['workclass', 'education','marital-status','occupation', 'relationship', 'ra
        print(f"\n\nUnique values of column {i} is:\n" , df[i].unique())
```

```


Unique values of column workclass is:
 ['State-gov' 'Self-emp-not-inc' 'Private' 'Federal-gov' 'Local-gov' '?'
 'Self-emp-inc' 'Without-pay' 'Never-worked']


Unique values of column education is:
 ['Bachelors' 'HS-grad' '11th' 'Masters' '9th' 'Some-college' 'Assoc-acdm'
 'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th'
 '1st-4th' 'Preschool' '12th']


Unique values of column marital-status is:
 ['Never-married' 'Married-civ-spouse' 'Divorced' 'Married-spouse-absent'
 'Separated' 'Married-AF-spouse' 'Widowed']


Unique values of column occupation is:
 ['Adm-clerical' 'Exec-managerial' 'Handlers-cleaners' 'Prof-specialty'
 'Other-service' 'Sales' 'Craft-repair' 'Transport-moving'
 'Farming-fishing' 'Machine-op-inspct' 'Tech-support' 'others'
 'Protective-serv' 'Armed-Forces' 'Priv-house-serv']


Unique values of column relationship is:
 ['Not-in-family' 'Husband' 'Wife' 'Own-child' 'Unmarried' 'Other-relativ
e']


Unique values of column race is:
 ['White' 'Black' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo' 'Other']


Unique values of column sex is:
 ['Male' 'Female']


Unique values of column native-country is:
 ['United-States' 'Cuba' 'Jamaica' 'India' 'some-country' 'Mexico' 'South'
 'Puerto-Rico' 'Honduras' 'England' 'Canada' 'Germany' 'Iran'
 'Philippines' 'Italy' 'Poland' 'Columbia' 'Cambodia' 'Thailand' 'Ecuador'
 'Laos' 'Taiwan' 'Haiti' 'Portugal' 'Dominican-Republic' 'El-Salvador'
 'France' 'Guatemala' 'China' 'Japan' 'Yugoslavia' 'Peru'
 'Outlying-US(Guam-USVI-etc)' 'Scotland' 'Trinadad&Tobago' 'Greece'
 'Nicaragua' 'Vietnam' 'Hong' 'Ireland' 'Hungary' 'Holand-Netherlands']


Unique values of column age_type is:
 ['adult', 'old', 'young']
Categories (3, object): ['young' < 'adult' < 'old']
```

In [41]:

```
df['workclass'].replace({'?':'Without-pay'},inplace=True)
```

In [42]:

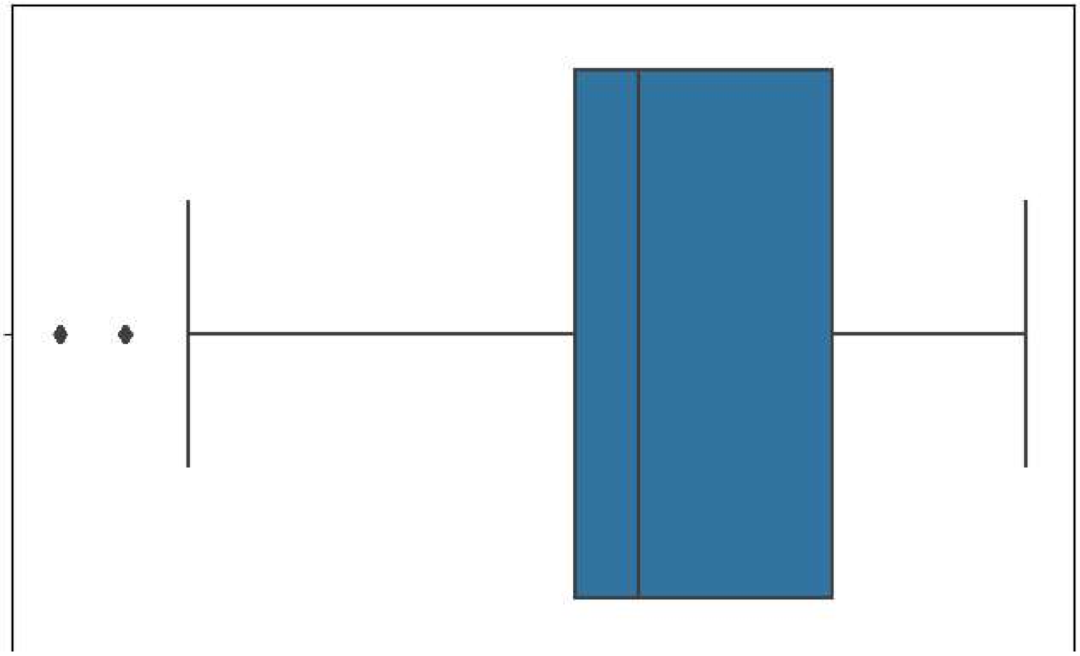
```
df.head(2)
```

Out[42]:

	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain
0	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0

In [43]:

```
for i in df.columns:
    if i in ['education-num','hours-per-week','capital-gain','capital-loss']:
        plt.subplots(figsize=(8,5))
        sns.boxplot(x=df[i],data=df)
```



In [44]:

```
abc=df[df['capital-gain']>25000].index
```

In [45]:

```
df.drop(index=abc,axis=0,inplace=True)
```

In [46]:

```
xyz=df[df['capital-loss']>3000].index
```

In [47]:

```
df.drop(index=xyz,axis=0,inplace=True)
```

In [48]:

```
df.shape
```

Out[48]:

```
(28871, 14)
```

In [ ]:

```

```

In [ ]:

```

```

In [49]:

```
df.describe().T
```

Out[49]:

	count	mean	std	min	25%	50%	75%	max
education-num	28871.0	10.082470	2.634933	1.0	9.0	10.0	13.0	16.0
capital-gain	28871.0	605.246510	2400.681346	0.0	0.0	0.0	0.0	22040.0
capital-loss	28871.0	96.485297	419.304715	0.0	0.0	0.0	0.0	2824.0
hours-per-week	28871.0	40.579474	12.714416	1.0	39.5	40.0	45.0	99.0
income_new	28871.0	0.242458	0.428577	0.0	0.0	0.0	0.0	1.0

In [50]:

```
df.head(2)
```

Out[50]:

	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain
0	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0

In [51]:

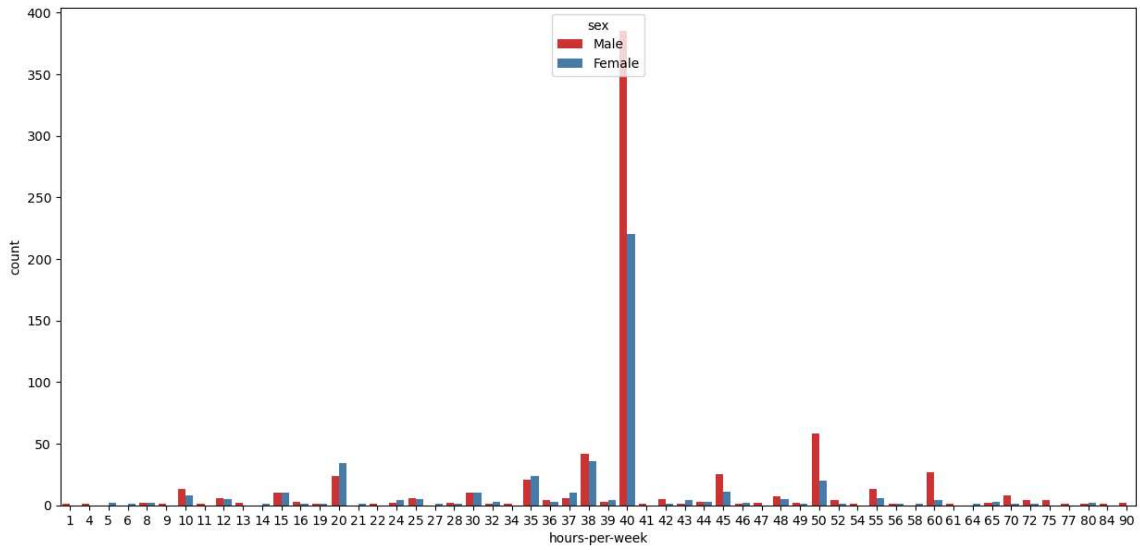
```
temp1=df[(df['workclass']=='State-gov') & (df['native-country']=='United-States')]
```

In [52]:

```
plt.figure(figsize=(15,7))
sns.countplot(x='hours-per-week',data=temp1,hue='sex',palette='Set1')
```

Out[52]:

```
<AxesSubplot:xlabel='hours-per-week', ylabel='count'>
```

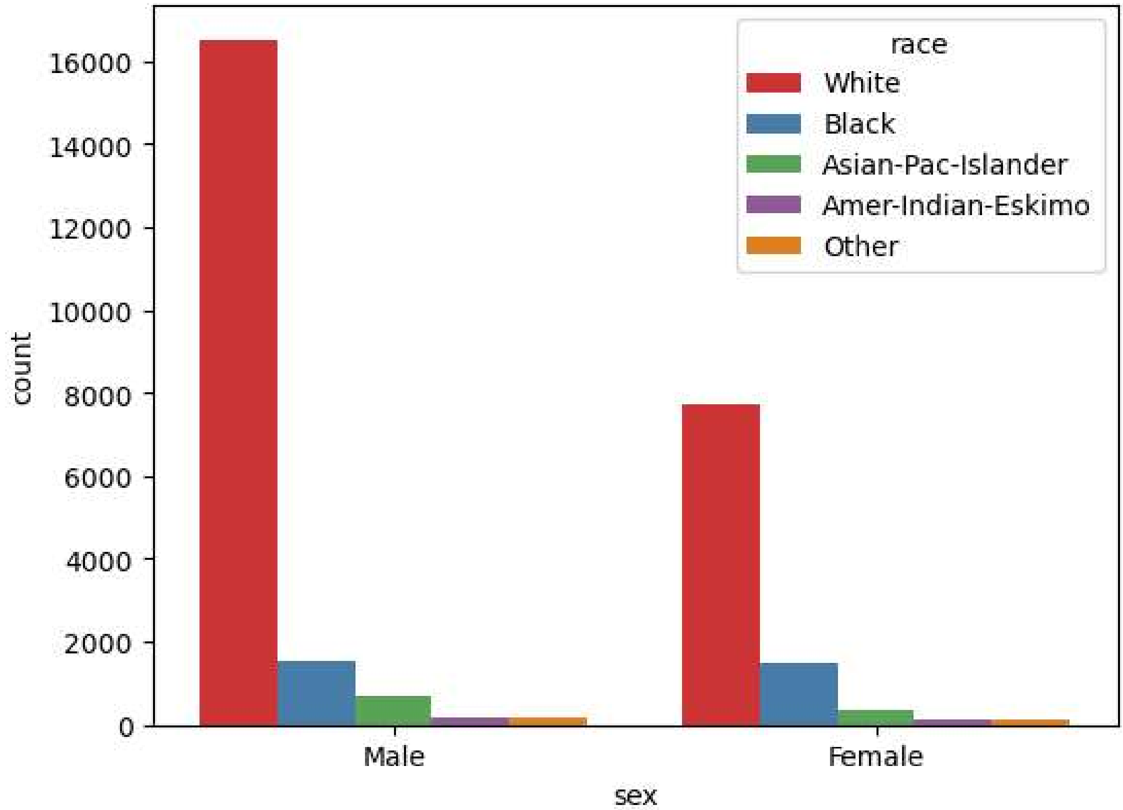


In [53]:

```
sns.countplot(x='sex',data=df,hue='race',palette='Set1')
```

Out[53]:

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

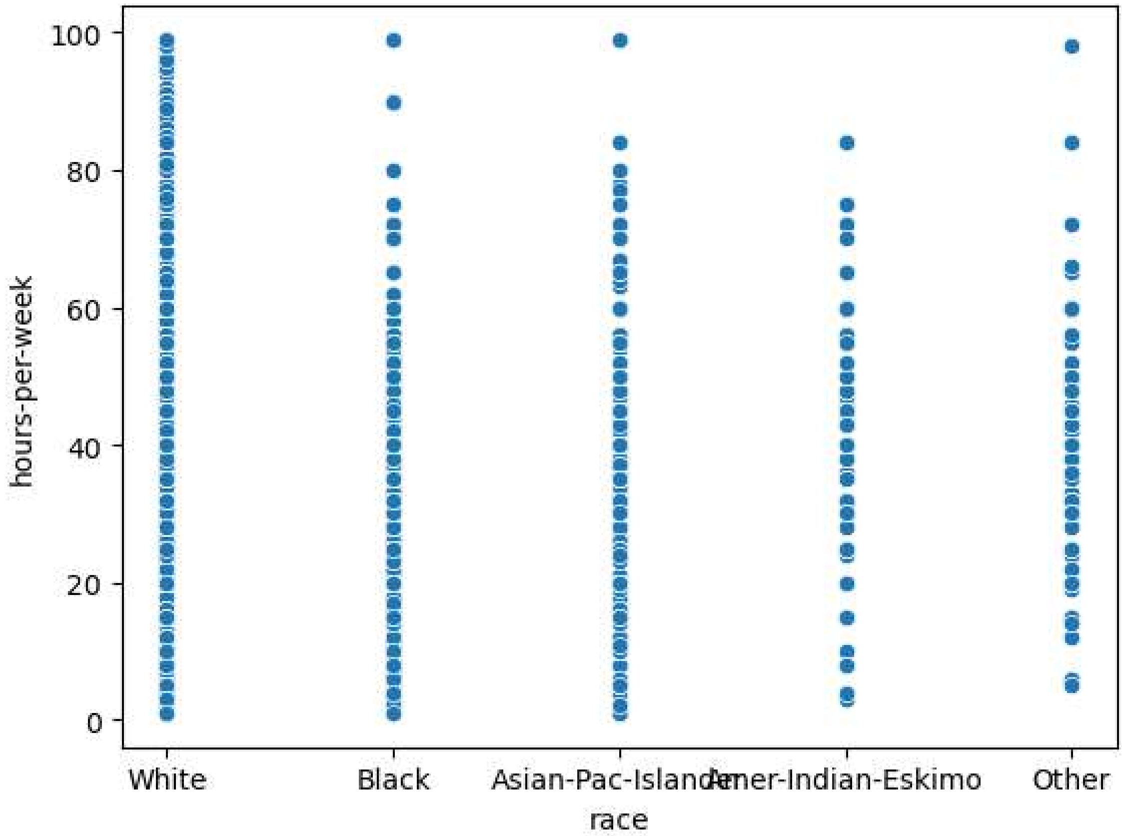


In [54]:

```
sns.scatterplot(x='race',y='hours-per-week',data=df)
```

Out[54]:

<AxesSubplot:xlabel='race', ylabel='hours-per-week'>



In [55]:

```
# sns.pairplot(data=df,hue='income_new')
# plt.show()
```

In [56]:

```
df.head(2)
```

Out[56]:

	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain
0	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174
1	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0

In [57]:

```
le=LabelEncoder()
```

In [58]:

```
for i in df.columns:
    if i not in ['education-num','capital-gain','capital-loss','hours-per-week']:
        df[i]=le.fit_transform(df[i])
```

In [59]:

```
df.head(5)
```

Out[59]:

	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	c
0	6	9	13	4	0	1	4	1	2174	
1	5	9	13	2	3	0	4	1	0	
2	3	11	9	0	5	1	4	1	0	
3	3	1	7	2	5	0	2	1	0	
4	3	9	13	2	9	5	2	0	0	

In [60]:

```
X=df.iloc[:,:-1].values
Y=df.iloc[:,13:].values
```

In [61]:

```
print(X.shape)
print(Y.shape)
```

```
(28871, 13)
(28871, 1)
```

In [62]:

```
scaler=MinMaxScaler()
```

In [63]:

```
X=scaler.fit_transform(X)
```

In [64]:

```
from sklearn.model_selection import train_test_split
```

In [65]:

```
X_train,X_test,y_train,y_test= train_test_split(X,Y,test_size=0.25)
```

In [66]:

```
print(X_train.shape)
print(y_train.shape)
```

```
(21653, 13)
(21653, 1)
```

In [67]:

```
print(X_test.shape)
print(y_test.shape)
```

```
(7218, 13)
(7218, 1)
```

Now implementing the Supervised learning models on by one

Implementing Logistic Regression Model

In [68]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,Confus
```

In [69]:

```
reg= LogisticRegression(fit_intercept=True,max_iter=100)
```

In [70]:

```
reg.fit(X_train,y_train)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:114
3: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[70]:

```
▾ LogisticRegression
LogisticRegression()
```

In [71]:

```
y_pred=reg.predict(X_test)
y_pred
```

Out[71]:

```
array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

In [72]:

```
print(reg.score(X_train,y_train))
print(reg.score(X_test,y_test))
```

```
0.8187318154528241
0.8194790800775839
```

In [73]:

```
acc_log=accuracy_score(y_pred,y_test)
acc_log
```

Out[73]:

```
0.8194790800775839
```

In [74]:

```
print(reg.coef_)
print('\n\n')
print(reg.intercept_)
```

```
[[-0.41147186  0.21973148  4.4118527  -1.4788101   0.08180961 -0.70712161
   0.62526581  0.85106368  7.19134536  1.94261265  2.02161838 -0.04085956
  -0.79494896]]



[-5.35564548]
```

In [75]:

```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.84      0.89      6221
           1       0.41      0.71      0.52       997

    accuracy                           0.82      7218
   macro avg       0.68      0.77      0.70      7218
weighted avg       0.87      0.82      0.84      7218
```

In [76]:

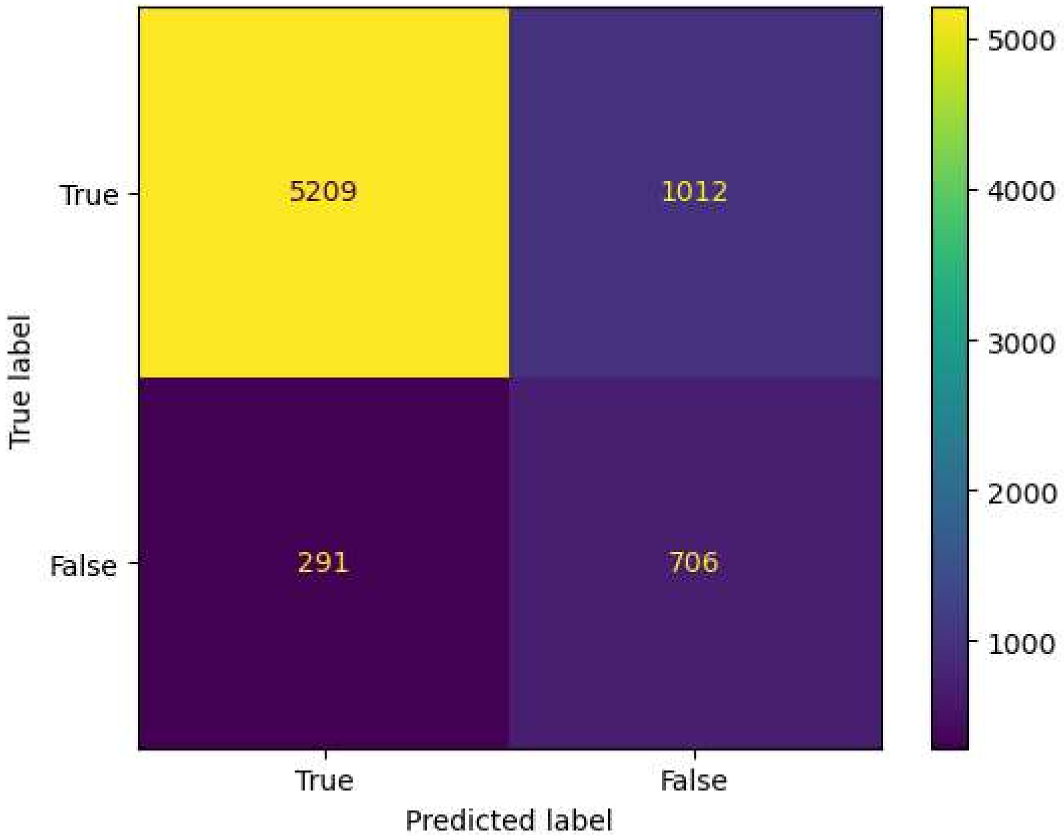
```
cf=confusion_matrix(y_pred,y_test)
cf
```

Out[76]:

```
array([[5209, 1012],
       [ 291,  706]], dtype=int64)
```

In [77]:

```
cnf=ConfusionMatrixDisplay(confusion_matrix=cf,display_labels=[True,False])
cnf.plot()
plt.show()
```



Using Naive Bayes Method

In [78]:

```
from sklearn.naive_bayes import GaussianNB
```

In [79]:

```
gauss= GaussianNB()
```

In [80]:

```
gauss.fit(X_train,y_train)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:114
3: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[80]:

```
▾ GaussianNB
GaussianNB()
```

In [81]:

```
y_pred=gauss.predict(X_test)
y_pred
```

Out[81]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [82]:

```
# if Income<=50K---->0 and >50 ---->1



acc_gauss=accuracy_score(y_pred,y_test)
acc_gauss
```

Out[82]:

```
0.8110279855915766
```

In [83]:

```
cf_gauss=confusion_matrix(y_pred,y_test)
cf_gauss
```

Out[83]:

```
array([[5104,  968],
       [ 396,  750]], dtype=int64)
```

Using Decision Tree Method

In [103]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
```

In [104]:

```
# Using Gini and finding the best fit as root node and other nodes


df= DecisionTreeClassifier(criterion='gini',max_depth=3)
```

In [105]:

```
df.fit(X_train,y_train)
```

Out[105]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

In [106]:

```
y_pred=df.predict(X_test)
```

In [107]:

```
# Using Entropy and finding the best fit as root node and other nodes

df2= DecisionTreeClassifier(criterion='entropy',max_depth=3)
```

In [108]:

```
df2.fit(X_train,y_train)
```

Out[108]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

In [109]:

```
y_pred2=df2.predict(X_test)
```

In [110]:

```
y_pred
```

Out[110]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [92]:

```
acc_df=accuracy_score(y_pred,y_test)
acc_df
```

Out[92]:

```
0.8320864505403158
```

In [111]:

```
acc_df2=accuracy_score(y_pred2,y_test)
acc_df2
```

Out[111]:

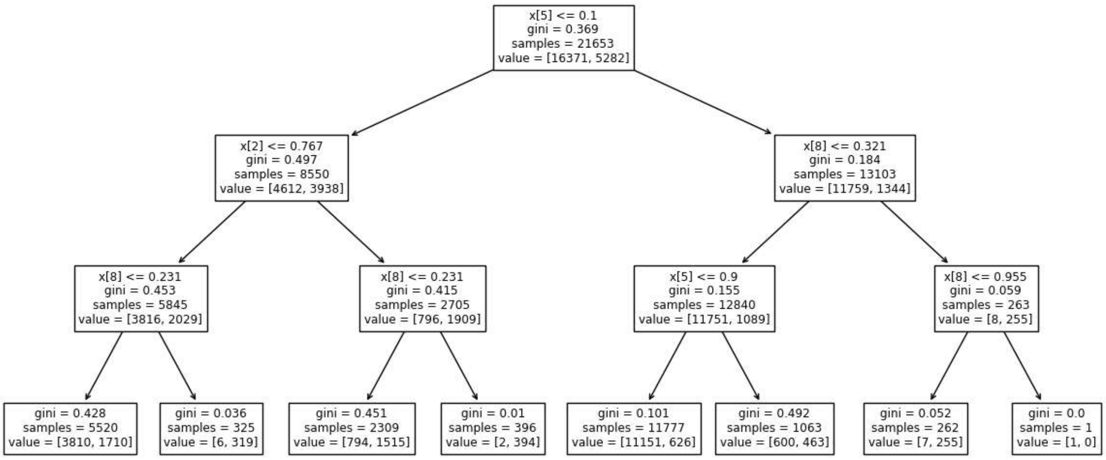
0.8345802161263508

In [96]:

```
plt.figure(figsize=(15,7))

plot_tree(df)

plt.plot()
plt.show()
```

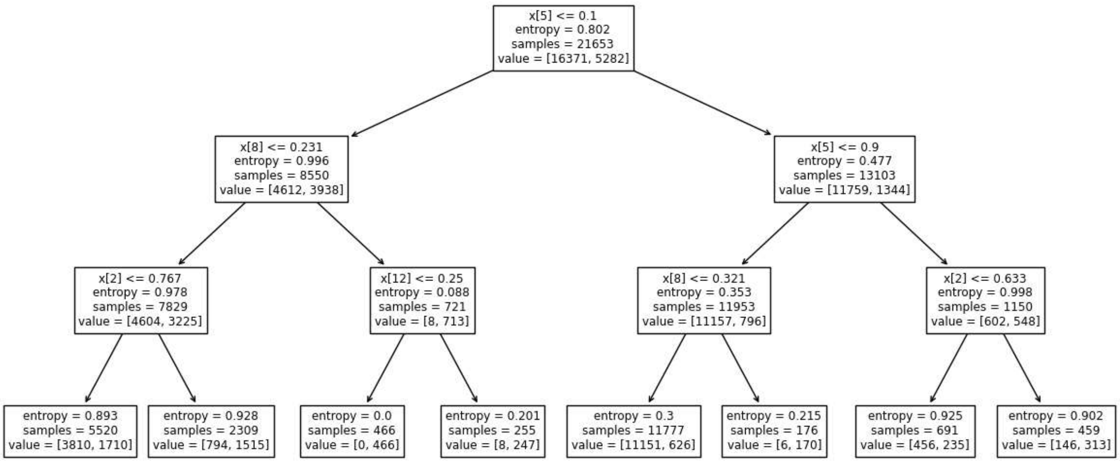


In [112]:

```
plt.figure(figsize=(15,7))

plot_tree(df2)

plt.plot()
plt.show()
```



Using Random Forest Method

In [113]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [124]:

```
rf= RandomForestClassifier(n_estimators=300,criterion='gini',max_depth=3,random_state=20
```

In [125]:

```
rf.fit(X_train,y_train)
```

```
C:\Users\dtdee\AppData\Local\Temp\ipykernel_19156\1593328843.py:1: DataCon
versionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  rf.fit(X_train,y_train)
```

Out[125]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=3, n_estimators=300, oob_score=True,
                       random_state=20)
```

In [126]:

```
y_pred= rf.predict(X_test)
y_pred
```

Out[126]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [130]:

```
acc_rf= accuracy_score(y_pred,y_test)
acc_rf
```

Out[130]:

```
0.8240509836519812
```

In [131]:

```
rf.oob_score_
```

Out[131]:

```
0.8199325728536462
```

Using SVM Model

In [141]:

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

In [156]:

```
svm=SVC( max_iter=100,kernel='rbf')
```

In [157]:

```
svm.fit(X_train,y_train)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:114
3: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: Conve
rgenceWarning: Solver terminated early (max_iter=100).  Consider pre-proce
ssing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
```

Out[157]:

```
▾ SVC
SVC(max_iter=100)
```

In [158]:

```
y_pred=svm.predict(X_test)
y_pred
```

Out[158]:

```
array([1, 1, 0, ..., 1, 1, 1], dtype=int64)
```

In [159]:

```
acc_svm= accuracy_score(y_pred,y_test)
acc_svm
```

Out[159]:

```
0.6546134663341646
```

Checking the Model Performance of all Models

In [162]:

```
pd.DataFrame({'Model Name':['LogisticRegression','NaiveBayes','Decision Tree','Random Fo
                 'Accuracy_score':[acc_log,acc_gauss,acc_df,acc_rf,acc_svm]},index=None)
```

Out[162]:

	Model Name	Accuracy_score
0	LogisticRegression	0.819479
1	NaiveBayes	0.811028
2	Decision Tree	0.832086
3	Random Forest	0.824051
4	SVM	0.654613

CONCLUSION:

Best Model for Classification would be Decision Tree and Random Forest

In [ ]:

```

```