

In this assignment students need to predict whether a person makes over 50K per year or not from classic adult dataset using XGBoost. The description of the dataset is as follows:

Data Set Information: Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Attribute Information: Listing of attributes:

50K, <=50K. age: continuous. workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. fnlwgt: continuous. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool. education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof- specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing,

Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces. relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. sex: Female, Male. capital-gain: continuous. capital-loss: continuous. hours-per-week: continuous. native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands. Following is the code to load required libraries and data: import numpy as np import pandas as pd

```
train_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning- (http://archive.ics.uci.edu/ml/machine-learning-) databases/adult/adult.data', header = None)
```

```
test_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning- (http://archive.ics.uci.edu/ml/machine-learning-) databases/adult/adult.test', skiprows = 1, header = None)
```

```
col_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation','relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'wage_class'] train_set.columns = col_labels test_set.columns = col_labels
```

```
In [1]: #code to load required libraries and data:
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

train_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data', header = None)

test_set = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test', skiprows = 1, header = None)

col_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status',
'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week',
'native_country', 'wage_class']
train_set.columns = col_labels
test_set.columns = col_labels
```

```
In [2]: train_set.shape,test_set.shape
```

```
Out[2]: ((32561, 15), (16281, 15))
```

```
In [3]: # View training and test data sample
train_set.sample(4, random_state = 42)
```

```
Out[3]:
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
14160	27	Private	160178	Some-college		10	Divorced	Adm-clerical
27048	45	State-gov	50567	HS-grad		9	Married-civ-spouse	Exec-managerial
28868	29	Private	185908	Bachelors		13	Married-civ-spouse	Exec-managerial
5667	30	Private	190040	Bachelors		13	Never-married	Machine-op-inspct



```
In [4]: test_set.sample(4, random_state = 42)
```

Out[4]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
13633	29	Private	189346	HS-grad		9	Never-married	Transport-moving
1921	31	Private	137076	Bachelors		13	Married-civ-spouse	Protective-serv
12140	52	Federal-gov	35546	HS-grad		9	Married-civ-spouse	Tech-support
9933	54	Local-gov	116428	10th		6	Married-civ-spouse	Exec-managerial

```
In [5]: # Checking null values in training and test data sets  
train_set.isnull().sum()
```

Out[5]:

```
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  
wage_class    0  
dtype: int64
```

```
In [6]: test_set.isnull().sum()
```

Out[6]:

```
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  
wage_class    0  
dtype: int64
```

```
In [7]: pd.DataFrame([train_set.dtypes, test_set.dtypes], index = ['train_set','test_set']).T
```

Out[7]:

	train_set	test_set
age	int64	int64
workclass	object	object
fnlwgt	int64	int64
education	object	object
education_num	int64	int64
marital_status	object	object
occupation	object	object
relationship	object	object
race	object	object
sex	object	object
capital_gain	int64	int64
capital_loss	int64	int64
hours_per_week	int64	int64
native_country	object	object
wage_class	object	object

```
In [8]: # Finding columns with data types as object
for i in train_set.columns:
    if train_set[i].dtypes == 'object':
        print(i)
```

workclass
education
marital_status
occupation
relationship
race
sex
native_country
wage_class

```
In [9]: train_set.workclass.value_counts()
```

```
Out[9]:
```

Private	22696
Self-emp-not-inc	2541
Local-gov	2093
?	1836
State-gov	1298
Self-emp-inc	1116
Federal-gov	960
Without-pay	14
Never-worked	7

Name: workclass, dtype: int64

In [10]: train_set

Out[10]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relations
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-fam
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-fam
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husba
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	W
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	W
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-fam
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husba
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-fam
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba
10	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husba
11	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husba
12	23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-cl
13	32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-fam
14	40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husba
15	34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husba
16	25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-cl
17	32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarri
18	38	Private	28887	11th	7	Married-civ-spouse	Sales	Husba
19	43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarri
20	40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husba
21	54	Private	302146	HS-grad	9	Separated	Other-service	Unmarri

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relations!
	22	35	Federal-gov	76845	9th	5	Married-civ-spouse	Farming-fishing
	23	43	Private	117037	11th	7	Married-civ-spouse	Transport-moving
	24	59	Private	109015	HS-grad	9	Divorced	Tech-support
	25	56	Local-gov	216851	Bachelors	13	Married-civ-spouse	Tech-support
	26	19	Private	168294	HS-grad	9	Never-married	Craft-repair
	27	54	?	180211	Some-college	10	Married-civ-spouse	?
	28	39	Private	367260	HS-grad	9	Divorced	Exec-managerial
	29	49	Private	193366	HS-grad	9	Married-civ-spouse	Craft-repair

32531	30	?	33811	Bachelors	13	Never-married	?	Not-in-farr
32532	34	Private	204461	Doctorate	16	Married-civ-spouse	Prof-specialty	Husba
32533	54	Private	337992	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba
32534	37	Private	179137	Some-college	10	Divorced	Adm-clerical	Unmarri
32535	22	Private	325033	12th	8	Never-married	Protective-serv	Own-cl
32536	34	Private	160216	Bachelors	13	Never-married	Exec-managerial	Not-in-farr
32537	30	Private	345898	HS-grad	9	Never-married	Craft-repair	Not-in-farr
32538	38	Private	139180	Bachelors	13	Divorced	Prof-specialty	Unmarri
32539	71	?	287372	Doctorate	16	Married-civ-spouse	?	Husba
32540	45	State-gov	252208	HS-grad	9	Separated	Adm-clerical	Own-cl
32541	41	?	202822	HS-grad	9	Separated	?	Not-in-farr
32542	72	?	129912	HS-grad	9	Married-civ-spouse	?	Husba
32543	45	Local-gov	119199	Assoc-acdm	12	Divorced	Prof-specialty	Unmarri
32544	31	Private	199655	Masters	14	Divorced	Other-service	Not-in-farr
32545	39	Local-gov	111499	Assoc-acdm	12	Married-civ-spouse	Adm-clerical	W

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
32546	37	Private	198216	Assoc-acdm	12	Divorced	Tech-support	Not-in-fam
32547	43	Private	260761	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husba
32548	65	Self-emp-not-inc	99359	Prof-school	15	Never-married	Prof-specialty	Not-in-fam
32549	43	State-gov	255835	Some-college	10	Divorced	Adm-clerical	Oth relat
32550	43	Self-emp-not-inc	27242	Some-college	10	Married-civ-spouse	Craft-repair	Husba
32551	32	Private	34066	10th	6	Married-civ-spouse	Handlers-cleaners	Husba
32552	43	Private	84661	Assoc-voc	11	Married-civ-spouse	Sales	Husba
32553	32	Private	116138	Masters	14	Never-married	Tech-support	Not-in-fam
32554	53	Private	321865	Masters	14	Married-civ-spouse	Exec-managerial	Husba
32555	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in-fam
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	W
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husba
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarri
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-ch
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	W

32561 rows × 15 columns



In [11]: `train_set.relationship.value_counts()`

Out[11]:

Husband	13193
Not-in-family	8305
Own-child	5068
Unmarried	3446
Wife	1568
Other-relative	981

Name: relationship, dtype: int64

```
In [12]: train_set.workclass.unique(),train_set.education.unique(),train_set.marital_status.unique(),train_set.native_country.unique()
```

```
Out[12]: (9, 16, 7, 42)
```

```
In [13]: X_train = train_set.copy()  
X_test = test_set.copy()
```

```
In [14]: X_train.columns
```

```
Out[14]: Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num',  
       'marital_status', 'occupation', 'relationship', 'race', 'sex',  
       'capital_gain', 'capital_loss', 'hours_per_week', 'native_country',  
       'wage_class'],  
      dtype='object')
```

```
In [15]: # Converting categorical values to numeric values  
dict_sex = {}  
count = 0  
for i in X_train.sex.unique():  
    dict_sex[i] = count  
    count +=1
```

```
In [16]: dict_workclass ={}  
count = 0  
for i in X_train.workclass.unique():  
    dict_workclass[i] = count  
    count +=1
```

```
In [17]: dict_education = {}
count = 0
for i in X_train.education.unique():
    dict_education[i] = count
    count +=1

dict_marital_status = {}
count = 0
for i in X_train.marital_status.unique():
    dict_marital_status[i] = count
    count +=1

dict_occupation = {}
count = 0
for i in X_train.occupation.unique():
    dict_occupation[i] = count
    count +=1
dict_relationship = {}
count = 0
for i in X_train.relationship.unique():
    dict_relationship[i] = count
    count +=1
dict_race = {}
count = 0
for i in X_train.race.unique():
    dict_race[i] = count
    count +=1

dict_native_country ={}
count = 0
for i in X_train.native_country.unique():
    dict_native_country[i] = count
    count +=1

dict_wage_class = {}
count = 0
for i in X_train.wage_class.unique():
    dict_wage_class[i] = count
    count +=1
```

```
In [18]: dict_sex,dict_education,dict_wage_class,dict_native_country,dict_race,dict_occ  
upation ,dict_marital_status
```

```
Out[18]: ({' Male': 0, ' Female': 1},
{' Bachelors': 0,
 ' HS-grad': 1,
 ' 11th': 2,
 ' Masters': 3,
 ' 9th': 4,
 ' Some-college': 5,
 ' Assoc-acdm': 6,
 ' Assoc-voc': 7,
 ' 7th-8th': 8,
 ' Doctorate': 9,
 ' Prof-school': 10,
 ' 5th-6th': 11,
 ' 10th': 12,
 ' 1st-4th': 13,
 ' Preschool': 14,
 ' 12th': 15},
{' <=50K': 0, ' >50K': 1},
{' United-States': 0,
 ' Cuba': 1,
 ' Jamaica': 2,
 ' India': 3,
 '?': 4,
 ' Mexico': 5,
 ' South': 6,
 ' Puerto-Rico': 7,
 ' Honduras': 8,
 ' England': 9,
 ' Canada': 10,
 ' Germany': 11,
 ' Iran': 12,
 ' Philippines': 13,
 ' Italy': 14,
 ' Poland': 15,
 ' Columbia': 16,
 ' Cambodia': 17,
 ' Thailand': 18,
 ' Ecuador': 19,
 ' Laos': 20,
 ' Taiwan': 21,
 ' Haiti': 22,
 ' Portugal': 23,
 ' Dominican-Republic': 24,
 ' El-Salvador': 25,
 ' France': 26,
 ' Guatemala': 27,
 ' China': 28,
 ' Japan': 29,
 ' Yugoslavia': 30,
 ' Peru': 31,
 ' Outlying-US(Guam-USVI-etc)': 32,
 ' Scotland': 33,
 ' Trinadad&Tobago': 34,
 ' Greece': 35,
 ' Nicaragua': 36,
 ' Vietnam': 37,
 ' Hong': 38},
```

```
'Ireland': 39,
'Hungary': 40,
'Holand-Netherlands': 41},
{'White': 0,
'Black': 1,
'Asian-Pac-Islander': 2,
'Amer-Indian-Eskimo': 3,
'Other': 4},
{'Adm-clerical': 0,
'Exec-managerial': 1,
'Handlers-cleaners': 2,
'Prof-specialty': 3,
'Other-service': 4,
'Sales': 5,
'Craft-repair': 6,
'Transport-moving': 7,
'Farming-fishing': 8,
'Machine-op-inspct': 9,
'Tech-support': 10,
?: 11,
'Protective-serv': 12,
'Armed-Forces': 13,
'Priv-house-serv': 14},
{'Never-married': 0,
'Married-civ-spouse': 1,
'Divorced': 2,
'Married-spouse-absent': 3,
'Separated': 4,
'Married-AF-spouse': 5,
'Widowed': 6})
```

```
In [19]: X_train['sex'] = X_train['sex'].map(dict_sex)
X_train['education'] = X_train['education'].map(dict_education)
X_train['wage_class'] = X_train['wage_class'].map(dict_wage_class)
X_train['native_country'] = X_train['native_country'].map(dict_native_country)
X_train['race'] = X_train['race'].map(dict_race)
X_train['occupation']=X_train['occupation'].map(dict_occupation)
X_train['marital_status'] = X_train['marital_status'].map(dict_marital_status)
X_train['workclass'] = X_train['workclass'].map(dict_workclass)
X_train['relationship'] = X_train['relationship'].map(dict_relationship)
```

```
In [20]: train.isnull().sum()
```

```
Out[20]: 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  
wage_class      0  
dtype: int64
```

```
In [21]: Xtrain = X_train.astype(int)
```

```
In [22]: X_train.head()
```

```
Out[22]:
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
0	39	0	77516	0	13	0	0	0
1	50	1	83311	0	13	1	1	1
2	38	2	215646	1	9	2	2	0
3	53	2	234721	2	7	1	2	1
4	28	2	338409	0	13	1	3	2

```
In [23]: X_train.describe()
```

```
Out[23]:
```

	age	workclass	fnlwgt	education	education_num	marital_status
count	32561.000000	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000
mean	38.581647	2.309972	1.897784e+05	3.424465	10.080679	1.083781
std	13.640433	1.225728	1.055500e+05	3.453582	2.572720	1.251381
min	17.000000	0.000000	1.228500e+04	0.000000	1.000000	0.000000
25%	28.000000	2.000000	1.178270e+05	1.000000	9.000000	0.000000
50%	37.000000	2.000000	1.783560e+05	2.000000	10.000000	1.000000
75%	48.000000	2.000000	2.370510e+05	5.000000	12.000000	1.000000
max	90.000000	8.000000	1.484705e+06	15.000000	16.000000	6.000000

```
In [24]: dict_wage_class = {}
count = 0
for i in X_test.wage_class.unique():
    dict_wage_class[i] = count
    count +=1

dict_native_country ={}
count = 0
for i in X_test.native_country.unique():
    dict_native_country[i] = count
    count +=1
```

```
In [25]: X_test['sex'] = X_test['sex'].map(dict_sex)
X_test['education'] = X_test['education'].map(dict_education)
X_test['wage_class'] = X_test['wage_class'].map(dict_wage_class)
X_test['native_country'] = X_test['native_country'].map(dict_native_country)
X_test['race'] = X_test['race'].map(dict_race)
X_test['occupation']=X_test['occupation'].map(dict_occupation)
X_test['marital_status'] = X_test['marital_status'].map(dict_marital_status)
X_test['workclass'] = X_test['workclass'].map(dict_workclass)
X_test['relationship'] = X_test['relationship'].map(dict_relationship)
```

```
In [26]: dict_wage_class
```

```
Out[26]: {'<=50K.': 0, '>50K.': 1}
```

```
In [27]: X_test.head()
```

```
Out[27]:
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship
0	25	2	226802	2	7	0	9	3
1	38	2	89814	1	9	1	8	1
2	28	4	336951	6	12	1	12	1
3	44	2	160323	5	10	1	9	1
4	18	5	103497	5	10	0	11	3



```
In [28]: X_test.describe()
```

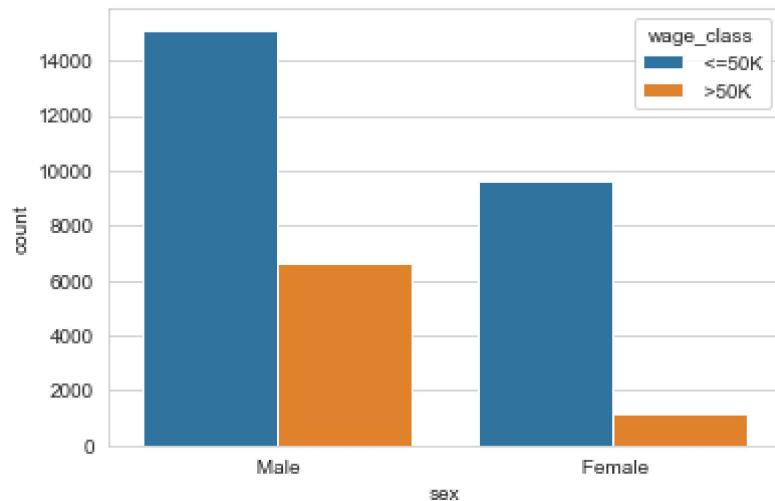
Out[28]:

	age	workclass	fnlwgt	education	education_num	marital_status
count	16281.000000	16281.000000	1.628100e+04	16281.000000	16281.000000	16281.000000
mean	38.767459	2.315030	1.894357e+05	3.386954	10.072907	1.084270
std	18.849187	1.246499	1.057149e+05	3.440725	2.567545	1.269622
min	1.000000	0.000000	1.349200e+04	0.000000	1.000000	0.000000
25%	28.000000	2.000000	1.167360e+05	1.000000	9.000000	0.000000
50%	37.000000	2.000000	1.778310e+05	2.000000	10.000000	1.000000
75%	48.000000	2.000000	2.383840e+05	5.000000	12.000000	1.000000
max	90.000000	8.000000	1.490400e+06	15.000000	16.000000	6.000000

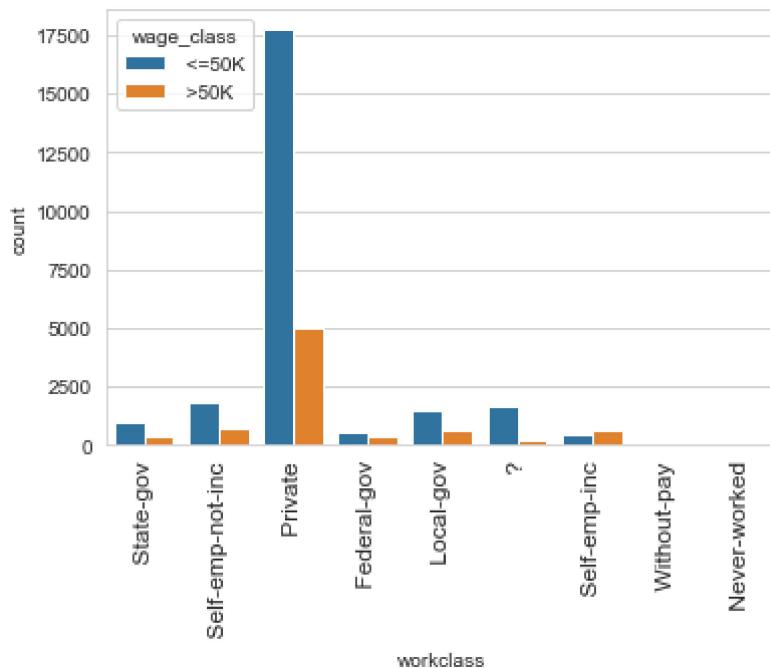
In [29]

```
# Performing annual income visual analysis
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(40,20))
sns.set_style('whitegrid')
%matplotlib inline
sns.countplot('sex', data=train_set, hue='wage_class')
```

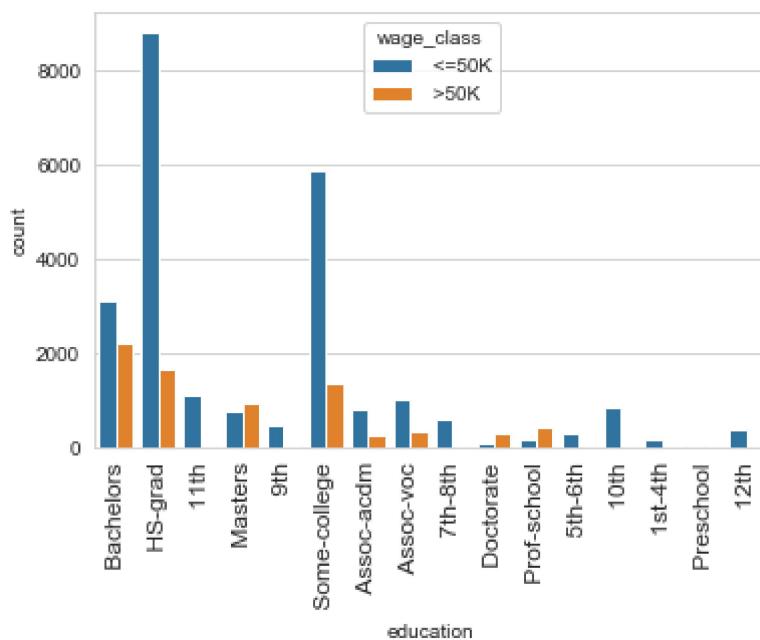
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9b5e9a90>



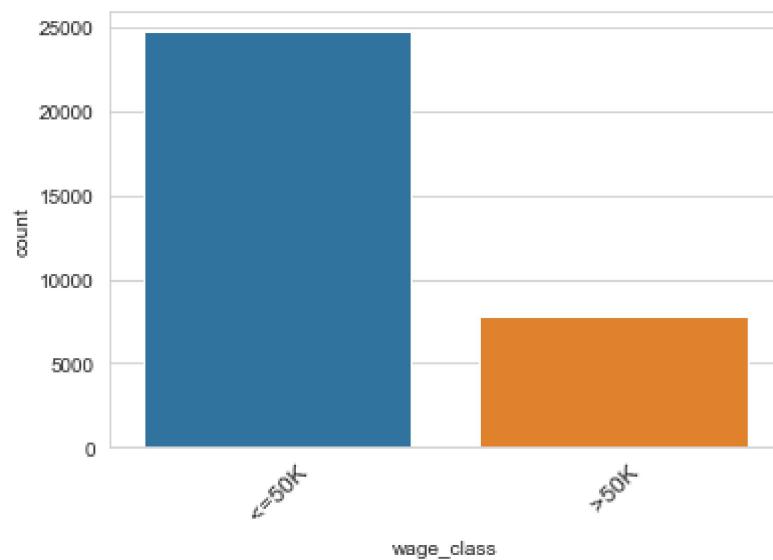
```
In [32]: g = sns.countplot('workclass',data=train_set,hue='wage_class')
g.set_xticklabels(g.get_xticklabels(), rotation = 90, fontsize = 12)
plt.show()
```



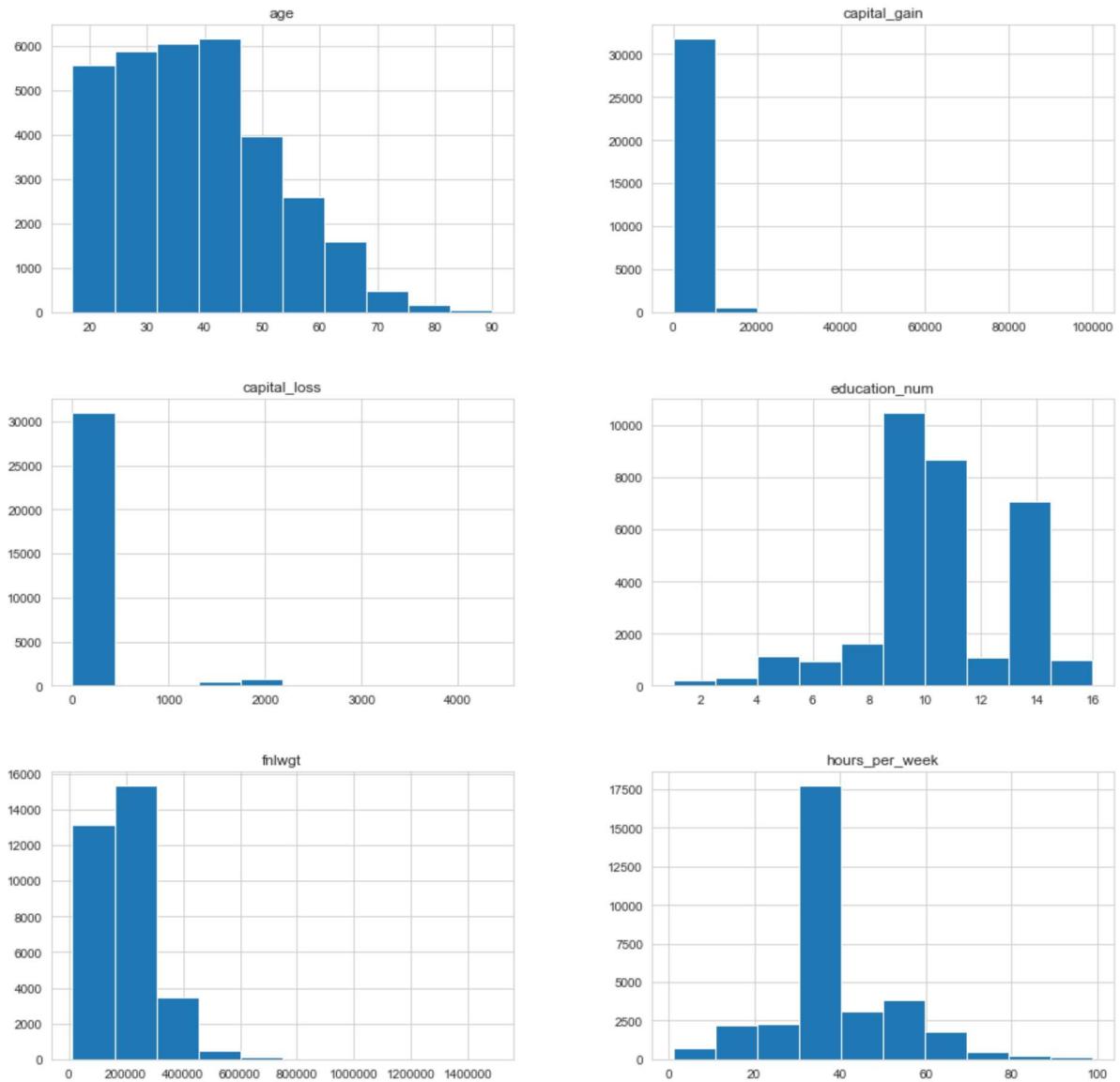
```
In [33]: g = sns.countplot('education',data=train_set,hue='wage_class')
g.set_xticklabels(g.get_xticklabels(), rotation = 90, fontsize = 12)
plt.show()
```



```
In [34]: g = sns.countplot('wage_class', data=train_set)
g.set_xticklabels(g.get_xticklabels(), rotation = 45, fontsize = 12)
plt.show()
```



```
In [35]: pd.DataFrame.hist(train_set,figsize = [15,15])  
plt.show()
```



```
In [36]: x_train = X_train.drop('wage_class',axis=1)  
y_train = X_train['wage_class']  
  
x_test = X_test.drop('wage_class',axis=1)  
y_test = X_test['wage_class']
```

```
In [37]: X = x_train.values  
Y = y_train.values  
Xtest = x_test.values  
Ytest = y_test.values
```

```
In [38]: x_train.shape,y_train.shape,X.shape,Y.shape,Xtest.shape,Ytest.shape
```

```
Out[38]: ((32561, 14), (32561,), (32561, 14), (32561,), (16281, 14), (16281,))
```

```
In [39]: Xtest
```

```
Out[39]: array([[ 25,      2, 226802, ...,     0,    40,     0],
   [ 38,      2, 89814, ...,     0,    50,     0],
   [ 28,      4, 336951, ...,     0,    40,     0],
   ...,
   [ 38,      2, 374983, ...,     0,    50,     0],
   [ 44,      2, 83891, ...,     0,    40,     0],
   [ 35,      6, 182148, ...,     0,    60,     0]], dtype=int64)
```

```
In [40]: # Using boosting method of ensemble model for predicting the annual income
```

```
from xgboost.sklearn import XGBClassifier
#set the parameters for the xgbosst model
params = {
    'objective': 'binary:logistic',
    'max_depth': 2,
    'learning_rate': 1.0,
    'silent': 1.0,
    'n_estimators': 5
}
params['eval_metric'] = ['logloss', 'auc']
```

```
In [41]: # Train the XGBClassifier model
bst = XGBClassifier(**params).fit(X,Y)
```

```
In [42]: # Predicting the annual income
preds = bst.predict(Xtest)
preds
```

```
Out[42]: array([0, 0, 0, ..., 1, 0, 1], dtype=int64)
```

```
In [43]: preds_proba = bst.predict_proba(Xtest)
preds_proba
```

```
Out[43]: array([[0.9862895 , 0.01371049],
   [0.6448917 , 0.35510832],
   [0.8749048 , 0.1250952 ],
   ...,
   [0.282      , 0.718      ],
   [0.71667016, 0.28332984],
   [0.17598617, 0.8240138 ]], dtype=float32)
```

```
In [44]: # Measure the accuracy of the model
correct = 0
from sklearn.metrics import accuracy_score
for i in range(len(preds)):
    if (y_test[i] == preds[i]):
        correct += 1

acc = accuracy_score(Ytest, preds)

print('Predicted correctly: {0}/{1}'.format(correct, len(preds)))
print('Accuracy Score :{:.4f}'.format(acc))
print('Error: {:.4f}'.format(1-acc))
```

Predicted correctly: 13897/16281

Accuracy Score :0.8536

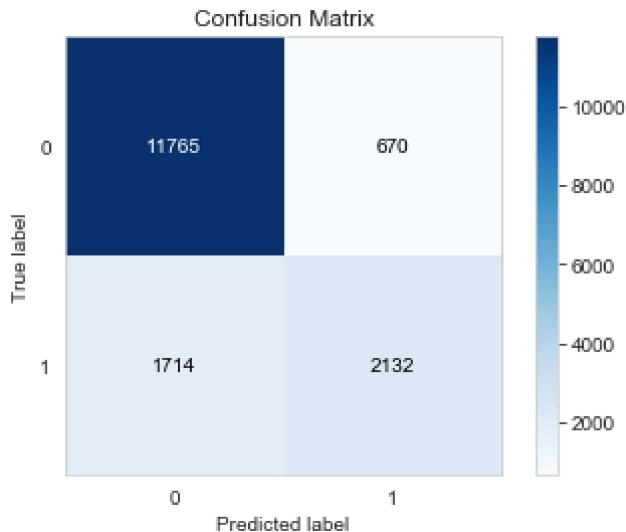
Error: 0.1464

```
In [45]: from sklearn.metrics import classification_report
print(classification_report(Ytest,preds))
```

	precision	recall	f1-score	support
0	0.87	0.95	0.91	12435
1	0.76	0.55	0.64	3846
micro avg	0.85	0.85	0.85	16281
macro avg	0.82	0.75	0.77	16281
weighted avg	0.85	0.85	0.85	16281

```
In [46]: # Creating confusion matrix
import scikitplot
scikitplot.metrics.plot_confusion_matrix(Ytest, preds)
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9d644860>

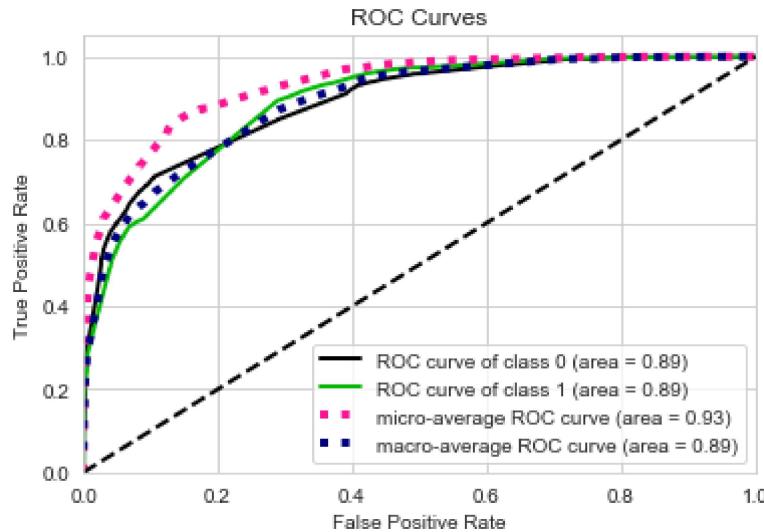


```
In [47]: #Creating ROC
```

```
scikitplot.metrics.plot_roc_curve(Ytest,preds_proba)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning: Function plot_roc_curve is deprecated; This will be removed in v0.5.0. Please use scikitplot.metrics.plot_roc instead.  
warnings.warn(msg, category=DeprecationWarning)
```

```
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9d6c8710>
```



```
In [48]: # Comparing the ensemble model with single random Logistic Regression model  
from sklearn.linear_model import LogisticRegression  
lr = LogisticRegression()  
lr.fit(X,Y)
```

```
Out[48]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
                           intercept_scaling=1, max_iter=100, multi_class='warn',  
                           n_jobs=None, penalty='l2', random_state=None, solver='warn',  
                           tol=0.0001, verbose=0, warm_start=False)
```

```
In [49]: pred_lr = lr.predict(Xtest)  
pred_lr
```

```
Out[49]: array([0, 0, 0, ..., 0, 1, 0], dtype=int64)
```

```
In [50]: pred_lr_proba = lr.predict_proba(Xtest)  
pred_lr_proba
```

```
Out[50]: array([[0.80955138, 0.19044862],  
                [0.71918966, 0.28081034],  
                [0.87925836, 0.12074164],  
                ...,  
                [0.89229153, 0.10770847],  
                [0.26855158, 0.73144842],  
                [0.79149081, 0.20850919]])
```

```
In [51]: print('Accuracy Score :{:.4f}'.format(accuracy_score(Ytest,pred_lr)))
```

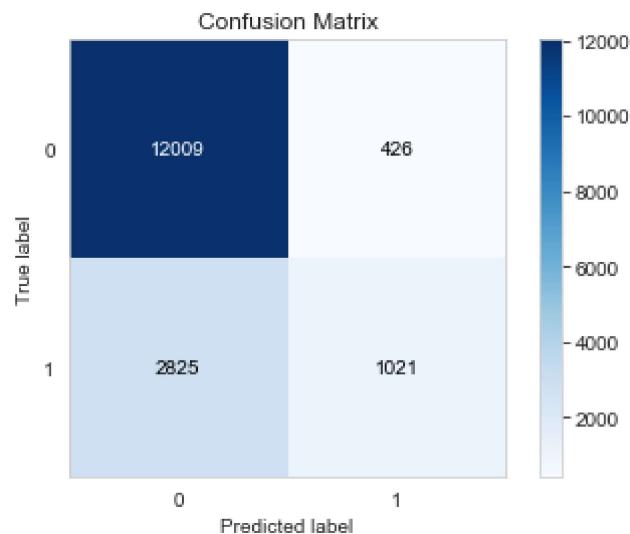
Accuracy Score :0.8003

```
In [52]: print(classification_report(Ytest,pred_lr))
```

	precision	recall	f1-score	support
0	0.81	0.97	0.88	12435
1	0.71	0.27	0.39	3846
micro avg	0.80	0.80	0.80	16281
macro avg	0.76	0.62	0.63	16281
weighted avg	0.79	0.80	0.76	16281

```
In [53]: #Confusion matrix
import scikitplot
scikitplot.metrics.plot_confusion_matrix(Ytest, pred_lr)
```

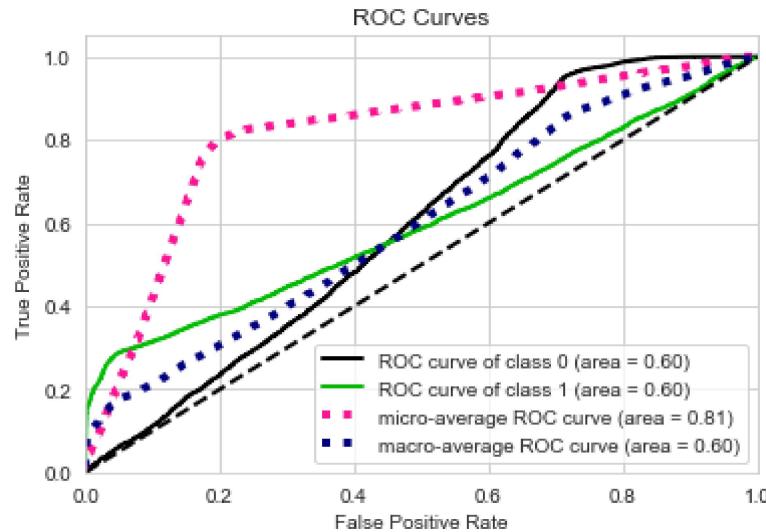
```
Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9d7573c8>
```



```
In [54]: # ROC  
scikitplot.metrics.plot_roc_curve(Ytest,pred_lr_proba)
```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning: Function plot_roc_curve is deprecated; This will be removed in v0.5.0. Please use scikitplot.metrics.plot_roc instead.
warnings.warn(msg, category=DeprecationWarning)

```
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9d7c4710>
```



```
In [55]: # Using Bagging Method of Ensemble model and base model as Logistic Regression  
  
from sklearn.ensemble import BaggingClassifier  
bag_LR = BaggingClassifier(LogisticRegression(),  
                           n_estimators=10, max_samples=0.5,  
                           bootstrap=True, random_state=3)
```

```
In [56]: bag_LR.fit(X,Y)
```

```
Out[56]: BaggingClassifier(base_estimator=LogisticRegression(C=1.0, class_weight=None,  
dual=False, fit_intercept=True,  
intercept_scaling=1, max_iter=100, multi_class='warn',  
n_jobs=None, penalty='l2', random_state=None, solver='warn',  
tol=0.0001, verbose=0, warm_start=False),  
bootstrap=True, bootstrap_features=False, max_features=1.0,  
max_samples=0.5, n_estimators=10, n_jobs=None, oob_score=False,  
random_state=3, verbose=0, warm_start=False)
```

```
In [57]: # Predictions by the Bagging Ensemble model  
bag_preds = bag_LR.predict(Xtest)  
bag_preds
```

```
Out[57]: array([0, 0, 0, ..., 0, 1, 0], dtype=int64)
```

```
In [58]: bag_preds_proba = bag_LR.predict_proba(Xtest)
bag_preds_proba
```

```
Out[58]: array([[0.81156283, 0.18843717],
 [0.73510814, 0.26489186],
 [0.87842468, 0.12157532],
 ...,
 [0.87698788, 0.12301212],
 [0.26852472, 0.73147528],
 [0.78501132, 0.21498868]])
```

```
In [59]: # Score of the bagging ensemble model
bag_LR.score(Xtest,Ytest)
```

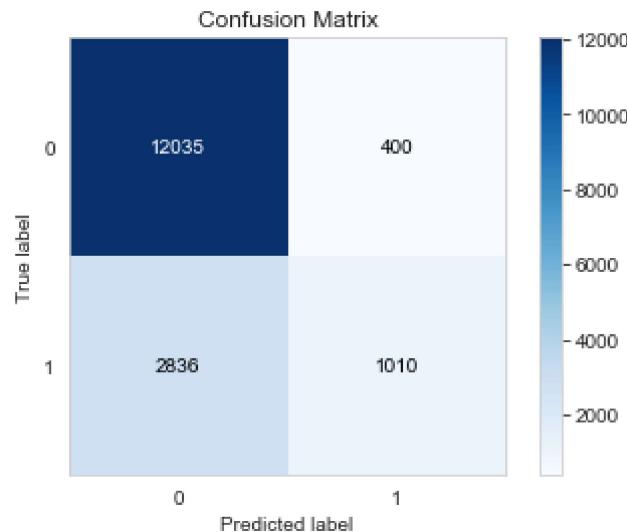
```
Out[59]: 0.8012407100300964
```

```
In [60]: print(accuracy_score(Ytest,bag_preds))
```

```
0.8012407100300964
```

```
In [61]: # Confusion matrix
scikitplot.metrics.plot_confusion_matrix(Ytest,bag_preds)
```

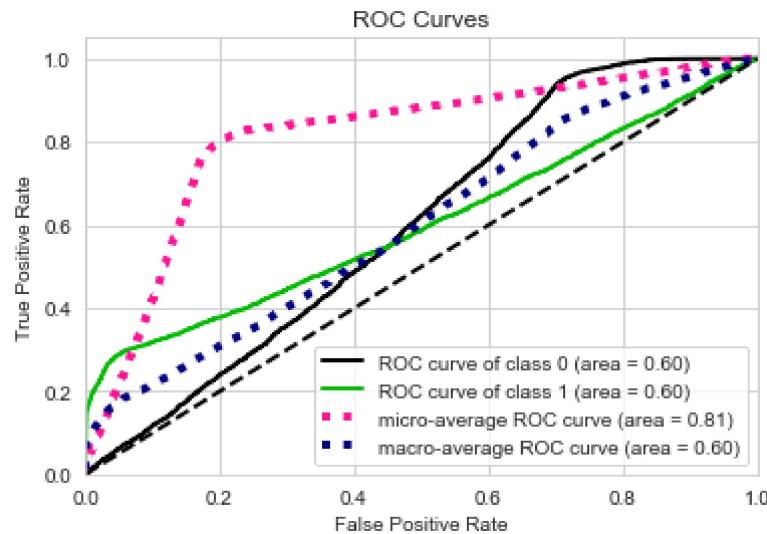
```
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9b8e5b70>
```



```
In [62]: # ROC  
scikitplot.metrics.plot_roc_curve(Ytest,bag_preds_proba)
```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning: Function plot_roc_curve is deprecated; This will be removed in v0.5.0. Please use scikitplot.metrics.plot_roc instead.
warnings.warn(msg, category=DeprecationWarning)

```
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x24a9b633978>
```



```
In [63]: # CONCLUSION  
# The boosting Classifier method yields a better performance in this scenario  
# as compared to single random Binary  
# classifier and bagging method.
```