In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('adult_data.csv'
                 , names=['age',
                          'workclass',
                          'fnlwgt',
                          'education',
                          'education-num',
                          'marital-status',
                          'occupation',
                          'relationship',
                          'race',
                          'sex',
                          'capital-gain',
                          'capital-loss',
                          'hours-per-week',
                          'native-country',
                          'income'
                         ]
                 , header=None)
df.head()
```

Out[2]:

	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	I
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	I
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	I
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	I
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fer

In [3]:
```
df.shape
```

Out[3]:
```
(32561, 15)
```

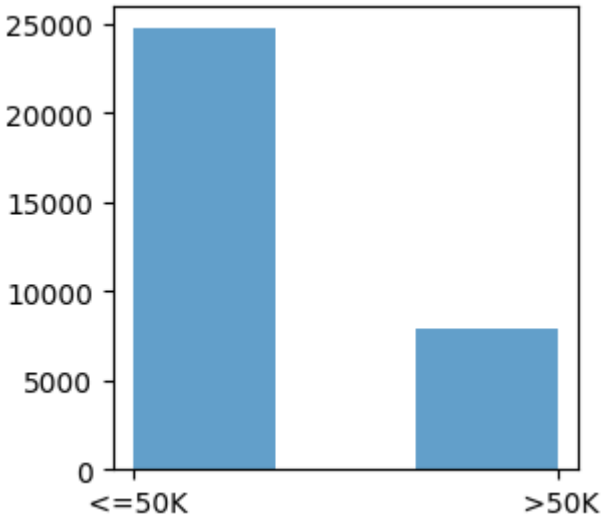
In [4]:
```
df['income'].unique()
```

Out[4]:
```
array([' <=50K', ' >50K'], dtype=object)
```

In [5]:
```
df['income'].value_counts()
```

Out[5]:
```
 <=50K    24720
 >50K      7841
Name: income, dtype: int64
```

In [6]:
```
plt.figure(figsize=(3,3))
plt.hist(df['income'],bins=3, alpha=0.7)
plt.show()
```



In [7]:
```
df.isnull().values.any()
```

Out[7]:
```
False
```

In [8]:
```
df['income'] = df['income'].map({' <=50K' :0, ' >50K' :1}).astype(int) #mapping nu
df.head()
```

Out[8]:

	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	I
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	I
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	I
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	I
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fer

In [9]:
```
df.describe()
```

Out[9]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	32
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000	
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456	
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429	
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000	
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000	
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000	
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000	
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000	

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
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
```

data description:

age: continuous.

workclass: Private, ?, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

it will be convert to new categories:

' Self-emp-not-inc' : 'self-employed',

' Self-emp-inc' : 'self-employed',

' Federal-gov' : 'gov', ' Local-gov' : 'gov', ' State-gov' : 'gov',

' Without-pay' : 'unemployed', ' Never-worked' : 'unemployed',

' Private' : ' Private',

' ?' : ' ?'.

fnlwgt (final weight): continuous. not neccesary. it will be droped.

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.

first of all it should be sorted: Preschool < 1st-4th < 5th-6th < 7th-8th < 9th < 10th < 11th < 12th < HS-grad < Prof-school < Assoc-acdm < Assoc-voc < Some-college < Bachelors < Masters < Doctorate

now, it will be convert to new categories:

' Preschool' :0, ' 1st-4th' :0, ' 5th-6th' :0, ' 7th-8th' :0, ' 9th' :0, ' 10th' :0, ' 11th' :0, ' 12th' :0,

' HS-grad' :1, ' Prof-school' :1,

' Assoc-acdm' :2, ' Assoc-voc' :2,

' Some-college' :3,

' Bachelors' :4,

' Masters' :5 ,

' Doctorate' :6

education-num: continuous. not neccesary. it will be droped.

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

it will be convert to new categories:

' Married-civ-spouse' :"married", ' Married-AF-spouse' :"married",

' Never-married' :"not-married", ' Divorced' :"not-married", ' Separated' :"not-married", ' Married-spouse-absent' :"not-married",

' Widowed' :"Widowed"

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.

it will be convert to new categories:

' Adm-clerical' :"adim", ' Exec-managerial' :"adim",

' Prof-specialty' :"Prof-specialty",

' Other-service' :"service", ' Protective-serv' :"service", ' Priv-house-serv' :"service",

' Sales' :"sales",

' Craft-repair' :"general", ' Transport-moving' :"general", ' Farming-fishing' :"general",

' Machine-op-inspct' :"general", ' Tech-support' :"general", ' Farming-fishing' :"general",

' Armed-Forces' :"military",

' ?':' ?'

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. not neccesary. it will be droped.

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.

In [11]:
```
df.drop(['fnlwgt'], axis=1, inplace=True)
df
```

Out[11]:

	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
...	...	...	...	...	...	...	...	...	...
32556	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female
32557	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male
32558	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female
32559	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male
32560	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female

32561 rows × 14 columns

In [12]:
```
df['workclass'].unique()
```

Out[12]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

In [13]:
```
df['workclass'] = df['workclass'].map({' Self-emp-not-inc' : 'self-employed', ' Sel
                                       ' Federal-gov' : 'gov', ' Local-gov' : 'gov
                                       ' Without-pay' : 'unemployed', ' Never-worke
                                       ' Private' : ' Private', ' ?' : ' ?'
                                      }).astype(str) #mapping numbers
df.head()
```

Out[13]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	cap
0	39	gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	
1	50	self-employed	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 [14]:
```
df['education'].unique()
```

Out[14]:
```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [15]:
```
df['education'] = df['education'].map({' Preschool' :0, ' 1st-4th' :0, ' 5th-6th'
df.head()
```

Out[15]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	cap
0	39	gov	4	13	Never-married	Adm-clerical	Not-in-family	White	Male	
1	50	self-employed	4	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	38	Private	1	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	53	Private	0	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	28	Private	4	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	

In [16]:
```
df.drop(['education-num'], axis=1, inplace=True)
df
```

Out[16]:

	age	workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	c
0	39	gov	4	Never-married	Adm-clerical	Not-in-family	White	Male	2174	
1	50	self-employed	4	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	
2	38	Private	1	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	
3	53	Private	0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	
4	28	Private	4	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	
...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	2	Married-civ-spouse	Tech-support	Wife	White	Female	0	
32557	40	Private	1	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	
32558	58	Private	1	Widowed	Adm-clerical	Unmarried	White	Female	0	
32559	22	Private	1	Never-married	Adm-clerical	Own-child	White	Male	0	
32560	52	self-employed	1	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	

32561 rows × 13 columns

In [17]:
```
df['marital-status'].unique()
```

Out[17]:
```
array([' Never-married', ' Married-civ-spouse', ' Divorced',
       ' Married-spouse-absent', ' Separated', ' Married-AF-spouse',
       ' Widowed'], dtype=object)
```

In [18]:
```
df['marital-status'] = df['marital-status'].map({' Married-civ-spouse' :"married",
df.head()
```

Out[18]:

	age	workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss
0	39	gov	4	not-married	Adm-clerical	Not-in-family	White	Male	2174	0
1	50	self-employed	4	married	Exec-managerial	Husband	White	Male	0	0
2	38	Private	1	not-married	Handlers-cleaners	Not-in-family	White	Male	0	0
3	53	Private	0	married	Handlers-cleaners	Husband	Black	Male	0	0
4	28	Private	4	married	Prof-specialty	Wife	Black	Female	0	0

In [19]:
```
df['occupation'].unique()
```

Out[19]:
```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Craft-repair',
       ' Transport-moving', ' Farming-fishing', ' Machine-op-inspct',
       ' Tech-support', ' ?', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

In [20]:
```
df['occupation'] = df['occupation'].map({' Adm-clerical' :"adim", ' Exec-managerial
                                         ' Prof-specialty' :"Prof-specialty",
                                         ' Other-service' :"service", ' Protective-
                                         ' Sales' :"sales",
                                         ' Craft-repair' :"general", ' Transport-mo
                                         ' Machine-op-inspct' :"general", ' Tech-su
                                         ' Armed-Forces' :"military",
                                         ' ?':' ?'
                                        }).astype(str) #mapping numbers
df.head()
```

Out[20]:

	age	workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss
0	39	gov	4	not-married	adim	Not-in-family	White	Male	2174	0
1	50	self-employed	4	married	adim	Husband	White	Male	0	0
2	38	Private	1	not-married	nan	Not-in-family	White	Male	0	0
3	53	Private	0	married	nan	Husband	Black	Male	0	0
4	28	Private	4	married	Prof-specialty	Wife	Black	Female	0	0

In [21]:
```
df.shape
```

Out[21]:
```
(32561, 13)
```

perfect! now data cleaning

In [22]:
```
df = df.replace(' \?',np.nan, regex=True)
```

In [23]:
```
df.dropna()
df.shape
```

Out[23]:
```
(32561, 13)
```

In [24]:
```
df['workclass'].unique()
```

Out[24]:
```
array(['gov', 'self-employed', ' Private', nan, 'unemployed'],
      dtype=object)
```

In [25]:
```
df.dropna(inplace=True)
df.shape
```

Out[25]:
```
(30162, 13)
```

In [26]:
```
df['workclass'].unique()
```

Out[26]:
```
array(['gov', 'self-employed', ' Private', 'unemployed'], dtype=object)
```

Data encoding

In [27]:
```
le = preprocessing.LabelEncoder()
df[[ 'sex']] = df[['sex']].apply(le.fit_transform)
df
```

Out[27]:

	age	workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	capit lo
0	39	gov	4	not-married	adim	Not-in-family	White	1	2174	
1	50	self-employed	4	married	adim	Husband	White	1	0	
2	38	Private	1	not-married	nan	Not-in-family	White	1	0	
3	53	Private	0	married	nan	Husband	Black	1	0	
4	28	Private	4	married	Prof-specialty	Wife	Black	0	0	
...	...	...	...	...	...	...	...	...	...	
32556	27	Private	2	married	general	Wife	White	0	0	
32557	40	Private	1	married	general	Husband	White	1	0	
32558	58	Private	1	Widowed	adim	Unmarried	White	0	0	
32559	22	Private	1	not-married	adim	Own-child	White	1	0	
32560	52	self-employed	1	married	adim	Wife	White	0	15024	

30162 rows × 13 columns

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

Out[28]:

	age	education	sex	capital-gain	capital-loss	hours-per-week	30
count	30162.000000	30162.000000	30162.000000	30162.000000	30162.000000	30162.000000	
mean	38.437902	2.175154	0.675685	1092.007858	88.372489	40.931238	
std	13.134665	1.544169	0.468126	7406.346497	404.298370	11.979984	
min	17.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
25%	28.000000	1.000000	0.000000	0.000000	0.000000	40.000000	
50%	37.000000	2.000000	1.000000	0.000000	0.000000	40.000000	
75%	47.000000	3.000000	1.000000	0.000000	0.000000	45.000000	
max	90.000000	6.000000	1.000000	99999.000000	4356.000000	99.000000	

In [29]:
```
df = pd.get_dummies(df)
df
```

Out[29]:

	age	education	sex	capital-gain	capital-loss	hours-per-week	income	workclass_Private	workclass_gov	work
0	39	4	1	2174	0	40	0	0	1	
1	50	4	1	0	0	13	0	0	0	
2	38	1	1	0	0	40	0	1	0	
3	53	0	1	0	0	40	0	1	0	
4	28	4	0	0	0	40	0	1	0	
...	...	...	...	...	...	...	...	...	...	
32556	27	2	0	0	0	38	0	1	0	
32557	40	1	1	0	0	40	1	1	0	
32558	58	1	0	0	0	40	0	1	0	
32559	22	1	1	0	0	20	0	1	0	
32560	52	1	0	15024	0	40	1	0	0	

30162 rows × 73 columns

In [30]:
```
#plt.close();
#sns.set_style("whitegrid");
#sns.pairplot(df, hue="income", height=3);
#plt.show()
```

In [31]:
```
df.columns
```

Out[31]:
```
Index(['age', 'education', 'sex', 'capital-gain', 'capital-loss',
       'hours-per-week', 'income', 'workclass_ Private', 'workclass_gov',
       'workclass_self-employed', 'workclass_unemployed',
       'marital-status_Widowed', 'marital-status_married',
       'marital-status_not-married', 'occupation_Prof-specialty',
       'occupation_adim', 'occupation_general', 'occupation_military',
       'occupation_nan', 'occupation_sales', 'occupation_service',
       'relationship_ Husband', 'relationship_ Not-in-family',
       'relationship_ Other-relative', 'relationship_ Own-child',
       'relationship_ Unmarried', 'relationship_ Wife',
       'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black',
       'race_ Other', 'race_ White', 'native-country_ Cambodia',
       'native-country_ Canada', 'native-country_ China',
       'native-country_ Columbia', 'native-country_ Cuba',
       'native-country_ Dominican-Republic', 'native-country_ Ecuador',
       'native-country_ El-Salvador', 'native-country_ England',
       'native-country_ France', 'native-country_ Germany',
       'native-country_ Greece', 'native-country_ Guatemala',
       'native-country_ Haiti', 'native-country_ Holand-Netherlands',
       'native-country_ Honduras', 'native-country_ Hong',
       'native-country_ Hungary', 'native-country_ India',
       'native-country_ Iran', 'native-country_ Ireland',
       'native-country_ Italy', 'native-country_ Jamaica',
       'native-country_ Japan', 'native-country_ Laos',
       'native-country_ Mexico', 'native-country_ Nicaragua',
       'native-country_ Outlying-US(Guam-USVI-etc)', 'native-country_ Peru',
       'native-country_ Philippines', 'native-country_ Poland',
       'native-country_ Portugal', 'native-country_ Puerto-Rico',
       'native-country_ Scotland', 'native-country_ South',
       'native-country_ Taiwan', 'native-country_ Thailand',
       'native-country_ Trinadad&Tobago', 'native-country_ United-States',
       'native-country_ Vietnam', 'native-country_ Yugoslavia'],
      dtype='object')
```

In [32]:
```
new_cols = ['age', 'education', 'sex', 'capital-gain', 'capital-loss',
       'hours-per-week', 'workclass_ Private', 'workclass_gov',
       'workclass_self-employed', 'workclass_unemployed',
       'marital-status_Widowed', 'marital-status_married',
       'marital-status_not-married', 'occupation_Prof-specialty',
       'occupation_adim', 'occupation_general', 'occupation_military',
       'occupation_nan', 'occupation_sales', 'occupation_service',
       'relationship_ Husband', 'relationship_ Not-in-family',
       'relationship_ Other-relative', 'relationship_ Own-child',
       'relationship_ Unmarried', 'relationship_ Wife',
       'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black',
       'race_ Other', 'race_ White', 'native-country_ Cambodia',
       'native-country_ Canada', 'native-country_ China',
       'native-country_ Columbia', 'native-country_ Cuba',
       'native-country_ Dominican-Republic', 'native-country_ Ecuador',
       'native-country_ El-Salvador', 'native-country_ England',
       'native-country_ France', 'native-country_ Germany',
       'native-country_ Greece', 'native-country_ Guatemala',
       'native-country_ Haiti', 'native-country_ Holand-Netherlands',
       'native-country_ Honduras', 'native-country_ Hong',
       'native-country_ Hungary', 'native-country_ India',
       'native-country_ Iran', 'native-country_ Ireland',
       'native-country_ Italy', 'native-country_ Jamaica',
       'native-country_ Japan', 'native-country_ Laos',
       'native-country_ Mexico', 'native-country_ Nicaragua',
       'native-country_ Outlying-US(Guam-USVI-etc)', 'native-country_ Peru',
       'native-country_ Philippines', 'native-country_ Poland',
```

```
       'native-country_ Portugal', 'native-country_ Puerto-Rico',
       'native-country_ Scotland', 'native-country_ South',
       'native-country_ Taiwan', 'native-country_ Thailand',
       'native-country_ Trinadad&Tobago', 'native-country_ United-States',
       'native-country_ Vietnam', 'native-country_ Yugoslavia', 'income']
df=df[new_cols]
```

In [33]:
```
df
```

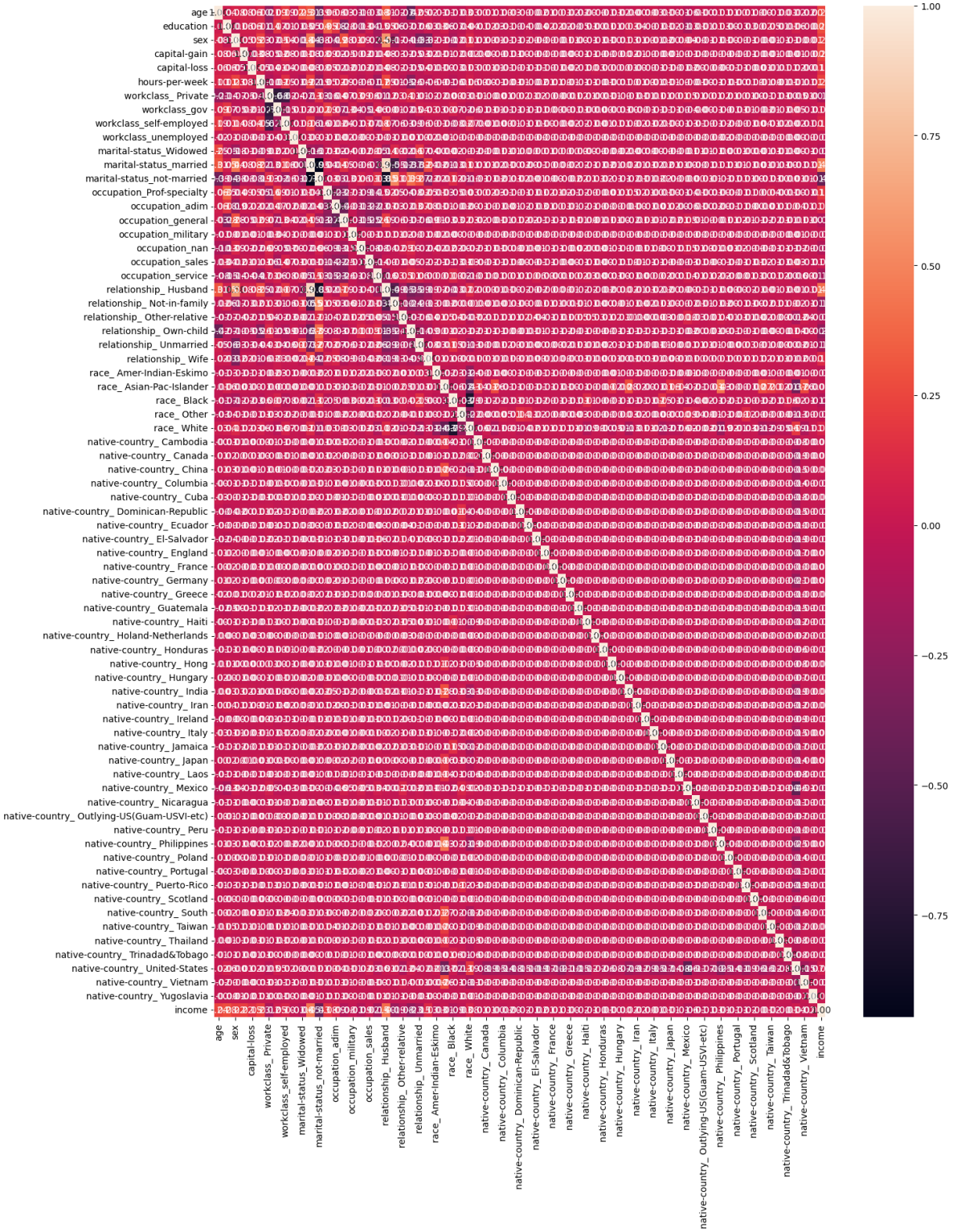
Out[33]:

	age	education	sex	capital-gain	capital-loss	hours-per-week	workclass_ Private	workclass_gov	workclass_se employe
0	39	4	1	2174	0	40	0	1	
1	50	4	1	0	0	13	0	0	
2	38	1	1	0	0	40	1	0	
3	53	0	1	0	0	40	1	0	
4	28	4	0	0	0	40	1	0	
...	...	...	...	...	...	...	...	...	
32556	27	2	0	0	0	38	1	0	
32557	40	1	1	0	0	40	1	0	
32558	58	1	0	0	0	40	1	0	
32559	22	1	1	0	0	20	1	0	
32560	52	1	0	15024	0	40	0	0	

30162 rows × 73 columns

plot correlation between attributes

In [34]:
```
from pylab import rcParams


def plot_correlation(data):
    '''
    plot correlation's matrix to explore dependency between features
    '''
    # init figure size
    rcParams['figure.figsize'] = 15, 20
    fig = plt.figure()
    sns.heatmap(data.corr(), annot=True, fmt=".2f")
    plt.show()
    fig.savefig('corr.png')


# plot correlation & densities
plot_correlation(df)
```



In [35]:
```
#plt.close();
#sns.set_style("whitegrid");
#sns.pairplot(df, hue="income", height=3);
#plt.show()
```

In [36]:
```
df.columns
```

Out[36]:
```
Index(['age', 'education', 'sex', 'capital-gain', 'capital-loss',
       'hours-per-week', 'workclass_ Private', 'workclass_gov',
       'workclass_self-employed', 'workclass_unemployed',
       'marital-status_Widowed', 'marital-status_married',
       'marital-status_not-married', 'occupation_Prof-specialty',
       'occupation_adim', 'occupation_general', 'occupation_military',
       'occupation_nan', 'occupation_sales', 'occupation_service',
       'relationship_ Husband', 'relationship_ Not-in-family',
       'relationship_ Other-relative', 'relationship_ Own-child',
       'relationship_ Unmarried', 'relationship_ Wife',
       'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black',
       'race_ Other', 'race_ White', 'native-country_ Cambodia',
       'native-country_ Canada', 'native-country_ China',
       'native-country_ Columbia', 'native-country_ Cuba',
       'native-country_ Dominican-Republic', 'native-country_ Ecuador',
       'native-country_ El-Salvador', 'native-country_ England',
       'native-country_ France', 'native-country_ Germany',
       'native-country_ Greece', 'native-country_ Guatemala',
       'native-country_ Haiti', 'native-country_ Holand-Netherlands',
       'native-country_ Honduras', 'native-country_ Hong',
       'native-country_ Hungary', 'native-country_ India',
       'native-country_ Iran', 'native-country_ Ireland',
       'native-country_ Italy', 'native-country_ Jamaica',
       'native-country_ Japan', 'native-country_ Laos',
       'native-country_ Mexico', 'native-country_ Nicaragua',
       'native-country_ Outlying-US(Guam-USVI-etc)', 'native-country_ Peru',
       'native-country_ Philippines', 'native-country_ Poland',
       'native-country_ Portugal', 'native-country_ Puerto-Rico',
       'native-country_ Scotland', 'native-country_ South',
       'native-country_ Taiwan', 'native-country_ Thailand',
       'native-country_ Trinadad&Tobago', 'native-country_ United-States',
       'native-country_ Vietnam', 'native-country_ Yugoslavia', 'income'],
      dtype='object')
```

In [37]:
```
df.shape
```

Out[37]:
```
(30162, 73)
```

In [38]:
```
X = df[['age', 'education', 'sex', 'capital-gain', 'capital-loss',
       'hours-per-week', 'workclass_ Private', 'workclass_gov',
       'workclass_self-employed', 'workclass_unemployed',
       'marital-status_Widowed', 'marital-status_married',
       'marital-status_not-married', 'occupation_Prof-specialty',
       'occupation_adim', 'occupation_general', 'occupation_military',
       'occupation_nan', 'occupation_sales', 'occupation_service',
       'relationship_ Husband', 'relationship_ Not-in-family',
       'relationship_ Other-relative', 'relationship_ Own-child',
       'relationship_ Unmarried', 'relationship_ Wife',
       'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black',
       'race_ Other', 'race_ White', 'native-country_ Cambodia',
       'native-country_ Canada', 'native-country_ China',
       'native-country_ Columbia', 'native-country_ Cuba',
       'native-country_ Dominican-Republic', 'native-country_ Ecuador',
       'native-country_ El-Salvador', 'native-country_ England',
       'native-country_ France', 'native-country_ Germany',
       'native-country_ Greece', 'native-country_ Guatemala',
       'native-country_ Haiti', 'native-country_ Holand-Netherlands',
       'native-country_ Honduras', 'native-country_ Hong',
       'native-country_ Hungary', 'native-country_ India',
       'native-country_ Iran', 'native-country_ Ireland',
```

```
       'native-country_ Italy', 'native-country_ Jamaica',
       'native-country_ Japan', 'native-country_ Laos',
       'native-country_ Mexico', 'native-country_ Nicaragua',
       'native-country_ Outlying-US(Guam-USVI-etc)', 'native-country_ Peru',
       'native-country_ Philippines', 'native-country_ Poland',
       'native-country_ Portugal', 'native-country_ Puerto-Rico',
       'native-country_ Scotland', 'native-country_ South',
       'native-country_ Taiwan', 'native-country_ Thailand',
       'native-country_ Trinadad&Tobago', 'native-country_ United-States',
       'native-country_ Vietnam', 'native-country_ Yugoslavia']].values
X[0:5]
```

Out[38]:
```
array([[  39,    4,    1, 2174,    0,   40,    0,    1,    0,    0,    0,
             0,    1,    0,    1,    0,    0,    0,    0,    0,    0,    1,
             0,    0,    0,    0,    0,    0,    0,    0,    1,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    1,    0,    0],
       [  50,    4,    1,    0,    0,   13,    0,    0,    1,    0,    0,
             1,    0,    0,    1,    0,    0,    0,    0,    0,    1,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    1,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    1,    0,    0],
       [  38,    1,    1,    0,    0,   40,    1,    0,    0,    0,    0,
             0,    1,    0,    0,    0,    0,    1,    0,    0,    0,    1,
             0,    0,    0,    0,    0,    0,    0,    0,    1,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    1,    0,    0],
       [  53,    0,    1,    0,    0,   40,    1,    0,    0,    0,    0,
             1,    0,    0,    0,    0,    0,    1,    0,    0,    1,    0,
             0,    0,    0,    0,    0,    0,    1,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    1,    0,    0],
       [  28,    4,    0,    0,    0,   40,    1,    0,    0,    0,    0,
             1,    0,    1,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    1,    0,    0,    1,    0,    0,    0,    0,
             0,    0,    1,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
             0,    0,    0,    0,    0,    0]])
```

In [39]:
```
Y = df['income'].values
Y[0:5]
```

Out[39]:
```
array([0, 0, 0, 0, 0])
```

standardizes features

In [40]:
```
sc = preprocessing.StandardScaler().fit(X)
X = sc.transform(X.astype(float))
X[0:5]
```

Out[40]:
```
array([[ 4.27957113e-02,  1.18178500e+00,  6.92806158e-01,
         1.46092284e-01, -2.18585975e-01, -7.77341106e-02,
        -1.68214415e+00,  2.45609619e+00, -3.66577248e-01,
        -2.15493793e-02, -1.67903484e-01, -9.36062492e-01,
         9.88921518e-01, -3.93154562e-01,  1.70603067e+00,
        -6.76457209e-01, -1.72765090e-02, -2.16461260e-01,
        -3.67217062e-01, -3.90959645e-01, -8.39144849e-01,
         1.70410113e+00, -1.74267846e-01, -4.16894915e-01,
        -3.45229885e-01, -2.21120202e-01, -9.78412046e-02,
        -1.74872860e-01, -3.20962622e-01, -8.78507304e-02,
         4.03824313e-01, -2.44363205e-02, -5.96669066e-02,
        -4.75351094e-02, -4.31288109e-02, -5.53129982e-02,
        -4.71835081e-02, -2.99327270e-02, -5.76754597e-02,
        -5.34735710e-02, -2.99327270e-02, -6.52827434e-02,
        -3.10225729e-02, -4.57503311e-02, -3.73419645e-02,
        -5.75807254e-03, -1.99501867e-02, -2.51063491e-02,
        -2.07651571e-02, -5.76754597e-02, -3.73419645e-02,
        -2.82194410e-02, -4.75351094e-02, -5.15693479e-02,
        -4.42711819e-02, -2.37474410e-02, -1.43671782e-01,
        -3.30951696e-02, -2.15493793e-02, -3.15534352e-02,
        -7.91966065e-02, -4.31288109e-02, -3.35934268e-02,
        -6.02239634e-02, -1.91005328e-02, -4.85747824e-02,
        -3.73419645e-02, -2.37474410e-02, -2.44363205e-02,
         3.10870534e-01, -4.61127654e-02, -2.30380196e-02],
       [ 8.80288144e-01,  1.18178500e+00,  6.92806158e-01,
        -1.47444622e-01, -2.18585975e-01, -2.33153070e+00,
        -1.68214415e+00, -4.07150178e-01,  2.72793799e+00,
        -2.15493793e-02, -1.67903484e-01,  1.06830474e+00,
        -1.01120259e+00, -3.93154562e-01,  1.70603067e+00,
        -6.76457209e-01, -1.72765090e-02, -2.16461260e-01,
        -3.67217062e-01, -3.90959645e-01,  1.19168937e+00,
        -5.86819634e-01, -1.74267846e-01, -4.16894915e-01,
        -3.45229885e-01, -2.21120202e-01, -9.78412046e-02,
        -1.74872860e-01, -3.20962622e-01, -8.78507304e-02,
         4.03824313e-01, -2.44363205e-02, -5.96669066e-02,
        -4.75351094e-02, -4.31288109e-02, -5.53129982e-02,
        -4.71835081e-02, -2.99327270e-02, -5.76754597e-02,
        -5.34735710e-02, -2.99327270e-02, -6.52827434e-02,
        -3.10225729e-02, -4.57503311e-02, -3.73419645e-02,
        -5.75807254e-03, -1.99501867e-02, -2.51063491e-02,
        -2.07651571e-02, -5.76754597e-02, -3.73419645e-02,
        -2.82194410e-02, -4.75351094e-02, -5.15693479e-02,
        -4.42711819e-02, -2.37474410e-02, -1.43671782e-01,
        -3.30951696e-02, -2.15493793e-02, -3.15534352e-02,
        -7.91966065e-02, -4.31288109e-02, -3.35934268e-02,
        -6.02239634e-02, -1.91005328e-02, -4.85747824e-02,
        -3.73419645e-02, -2.37474410e-02, -2.44363205e-02,
         3.10870534e-01, -4.61127654e-02, -2.30380196e-02],
       [-3.33399643e-02, -7.61039393e-01,  6.92806158e-01,
        -1.47444622e-01, -2.18585975e-01, -7.77341106e-02,
         5.94479374e-01, -4.07150178e-01, -3.66577248e-01,
        -2.15493793e-02, -1.67903484e-01, -9.36062492e-01,
         9.88921518e-01, -3.93154562e-01, -5.86155934e-01,
        -6.76457209e-01, -1.72765090e-02,  4.61976430e+00,
        -3.67217062e-01, -3.90959645e-01, -8.39144849e-01,
         1.70410113e+00, -1.74267846e-01, -4.16894915e-01,
        -3.45229885e-01, -2.21120202e-01, -9.78412046e-02,
        -1.74872860e-01, -3.20962622e-01, -8.78507304e-02,
         4.03824313e-01, -2.44363205e-02, -5.96669066e-02,
        -4.75351094e-02, -4.31288109e-02, -5.53129982e-02,
        -4.71835081e-02, -2.99327270e-02, -5.76754597e-02,
```

```
       -5.34735710e-02, -2.99327270e-02, -6.52827434e-02,
       -3.10225729e-02, -4.57503311e-02, -3.73419645e-02,
       -5.75807254e-03, -1.99501867e-02, -2.51063491e-02,
       -2.07651571e-02, -5.76754597e-02, -3.73419645e-02,
       -2.82194410e-02, -4.75351094e-02, -5.15693479e-02,
       -4.42711819e-02, -2.37474410e-02, -1.43671782e-01,
       -3.30951696e-02, -2.15493793e-02, -3.15534352e-02,
       -7.91966065e-02, -4.31288109e-02, -3.35934268e-02,
       -6.02239634e-02, -1.91005328e-02, -4.85747824e-02,
       -3.73419645e-02, -2.37474410e-02, -2.44363205e-02,
        3.10870534e-01, -4.61127654e-02, -2.30380196e-02],
      [ 1.10869517e+00, -1.40864752e+00,  6.92806158e-01,
       -1.47444622e-01, -2.18585975e-01, -7.77341106e-02,
        5.94479374e-01, -4.07150178e-01, -3.66577248e-01,
       -2.15493793e-02, -1.67903484e-01,  1.06830474e+00,
       -1.01120259e+00, -3.93154562e-01, -5.86155934e-01,
       -6.76457209e-01, -1.72765090e-02,  4.61976430e+00,
       -3.67217062e-01, -3.90959645e-01,  1.19168937e+00,
       -5.86819634e-01, -1.74267846e-01, -4.16894915e-01,
       -3.45229885e-01, -2.21120202e-01, -9.78412046e-02,
       -1.74872860e-01,  3.11562759e+00, -8.78507304e-02,
       -2.47632440e+00, -2.44363205e-02, -5.96669066e-02,
       -4.75351094e-02, -4.31288109e-02, -5.53129982e-02,
       -4.71835081e-02, -2.99327270e-02, -5.76754597e-02,
       -5.34735710e-02, -2.99327270e-02, -6.52827434e-02,
       -3.10225729e-02, -4.57503311e-02, -3.73419645e-02,
       -5.75807254e-03, -1.99501867e-02, -2.51063491e-02,
       -2.07651571e-02, -5.76754597e-02, -3.73419645e-02,
       -2.82194410e-02, -4.75351094e-02, -5.15693479e-02,
       -4.42711819e-02, -2.37474410e-02, -1.43671782e-01,
       -3.30951696e-02, -2.15493793e-02, -3.15534352e-02,
       -7.91966065e-02, -4.31288109e-02, -3.35934268e-02,
       -6.02239634e-02, -1.91005328e-02, -4.85747824e-02,
       -3.73419645e-02, -2.37474410e-02, -2.44363205e-02,
        3.10870534e-01, -4.61127654e-02, -2.30380196e-02],
      [-7.94696721e-01,  1.18178500e+00, -1.44340518e+00,
       -1.47444622e-01, -2.18585975e-01, -7.77341106e-02,
        5.94479374e-01, -4.07150178e-01, -3.66577248e-01,
       -2.15493793e-02, -1.67903484e-01,  1.06830474e+00,
       -1.01120259e+00,  2.54352892e+00, -5.86155934e-01,
       -6.76457209e-01, -1.72765090e-02, -2.16461260e-01,
       -3.67217062e-01, -3.90959645e-01, -8.39144849e-01,
       -5.86819634e-01, -1.74267846e-01, -4.16894915e-01,
       -3.45229885e-01,  4.52242712e+00, -9.78412046e-02,
       -1.74872860e-01,  3.11562759e+00, -8.78507304e-02,
       -2.47632440e+00, -2.44363205e-02, -5.96669066e-02,
       -4.75351094e-02, -4.31288109e-02,  1.80789332e+01,
       -4.71835081e-02, -2.99327270e-02, -5.76754597e-02,
       -5.34735710e-02, -2.99327270e-02, -6.52827434e-02,
       -3.10225729e-02, -4.57503311e-02, -3.73419645e-02,
       -5.75807254e-03, -1.99501867e-02, -2.51063491e-02,
       -2.07651571e-02, -5.76754597e-02, -3.73419645e-02,
       -2.82194410e-02, -4.75351094e-02, -5.15693479e-02,
       -4.42711819e-02, -2.37474410e-02, -1.43671782e-01,
       -3.30951696e-02, -2.15493793e-02, -3.15534352e-02,
       -7.91966065e-02, -4.31288109e-02, -3.35934268e-02,
       -6.02239634e-02, -1.91005328e-02, -4.85747824e-02,
       -3.73419645e-02, -2.37474410e-02, -2.44363205e-02,
       -3.21677320e+00, -4.61127654e-02, -2.30380196e-02]])
```

splitting data for train and test

In [41]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_sta
print('train set : ', x_train.shape, y_train.shape)
print('test set : ', x_test.shape, y_test.shape)
```

```
train set :  (24129, 72) (24129,)
test set :  (6033, 72) (6033,)
```

In [42]:
```
from sklearn.preprocessing import StandardScaler
```

In [43]:
```
scaler = StandardScaler().fit(x_train)
x_train_sc = scaler.transform(x_train)

x_test_sc = scaler.transform(x_test)
x_train_sc[0:5]
```

Out[43]:
```
array([[ 7.29633899e-01, -1.41112604e+00,  6.92124563e-01,
        -1.49654480e-01, -2.19627094e-01,  2.54264190e-01,
         5.94246755e-01, -4.08317376e-01, -3.65096917e-01,
        -2.13562845e-02, -1.66780219e-01,  1.07120139e+00,
        -1.01465417e+00, -3.91866439e-01, -5.84863867e-01,
         1.47573552e+00, -1.70350088e-02, -2.17406589e-01,
        -3.66626061e-01, -3.92358867e-01,  1.19476558e+00,
        -5.88693059e-01, -1.75000915e-01, -4.17325513e-01,
        -3.44211323e-01, -2.21349813e-01, -9.61445151e-02,
        -1.75000915e-01, -3.22880042e-01, -8.71786495e-02,
         4.04856915e-01, -2.49408357e-02, -5.94574259e-02,
        -4.77977239e-02, -4.27418202e-02, -5.62111112e-02,
        -4.55686097e-02, -2.95140668e-02, -5.83951091e-02,
        -5.35521281e-02, -2.88021826e-02, -6.61107357e-02,
        -3.15538279e-02, -4.46460827e-02, -3.81135647e-02,
        -6.43782753e-03, -1.82115680e-02, -2.03619962e-02,
        -2.23063741e-02, -5.94574259e-02, -3.86550118e-02,
        -3.02092370e-02, -4.73602223e-02, -5.15699908e-02,
        -4.69186781e-02, -2.40946368e-02, -1.46059349e-01,
        -3.28436279e-02, -2.13562845e-02, -3.34699714e-02,
        -7.66677179e-02, -4.32256913e-02, -3.52826435e-02,
        -5.94574259e-02, -1.93166852e-02, -4.95094339e-02,
        -3.58666114e-02, -1.82115680e-02, -2.40946368e-02,
         3.12443417e-01, -4.77977239e-02, -2.32176916e-02],
       [ 1.41502796e+00,  2.46631022e+00,  6.92124563e-01,
        -1.49654480e-01,  4.65776107e+00, -7.80026269e-02,
         5.94246755e-01, -4.08317376e-01, -3.65096917e-01,
        -2.13562845e-02, -1.66780219e-01,  1.07120139e+00,
        -1.01465417e+00, -3.91866439e-01, -5.84863867e-01,
         1.47573552e+00, -1.70350088e-02, -2.17406589e-01,
        -3.66626061e-01, -3.92358867e-01,  1.19476558e+00,
        -5.88693059e-01, -1.75000915e-01, -4.17325513e-01,
        -3.44211323e-01, -2.21349813e-01, -9.61445151e-02,
        -1.75000915e-01, -3.22880042e-01, -8.71786495e-02,
         4.04856915e-01, -2.49408357e-02, -5.94574259e-02,
        -4.77977239e-02, -4.27418202e-02, -5.62111112e-02,
        -4.55686097e-02, -2.95140668e-02, -5.83951091e-02,
        -5.35521281e-02, -2.88021826e-02,  1.51261363e+01,
        -3.15538279e-02, -4.46460827e-02, -3.81135647e-02,
        -6.43782753e-03, -1.82115680e-02, -2.03619962e-02,
        -2.23063741e-02, -5.94574259e-02, -3.86550118e-02,
        -3.02092370e-02, -4.73602223e-02, -5.15699908e-02,
        -4.69186781e-02, -2.40946368e-02, -1.46059349e-01,
        -3.28436279e-02, -2.13562845e-02, -3.34699714e-02,
        -7.66677179e-02, -4.32256913e-02, -3.52826435e-02,
        -5.94574259e-02, -1.93166852e-02, -4.95094339e-02,
        -3.58666114e-02, -1.82115680e-02, -2.40946368e-02,
        -3.20057951e+00, -4.77977239e-02, -2.32176916e-02],
       [-5.64999319e-01, -7.64886667e-01,  6.92124563e-01,
        -1.49654480e-01, -2.19627094e-01, -7.80026269e-02,
         5.94246755e-01, -4.08317376e-01, -3.65096917e-01,
        -2.13562845e-02, -1.66780219e-01,  1.07120139e+00,
        -1.01465417e+00, -3.91866439e-01, -5.84863867e-01,
         1.47573552e+00, -1.70350088e-02, -2.17406589e-01,
        -3.66626061e-01, -3.92358867e-01,  1.19476558e+00,
        -5.88693059e-01, -1.75000915e-01, -4.17325513e-01,
        -3.44211323e-01, -2.21349813e-01, -9.61445151e-02,
        -1.75000915e-01, -3.22880042e-01, -8.71786495e-02,
         4.04856915e-01, -2.49408357e-02, -5.94574259e-02,
        -4.77977239e-02, -4.27418202e-02, -5.62111112e-02,
        -4.55686097e-02, -2.95140668e-02, -5.83951091e-02,
```

```
       -5.35521281e-02, -2.88021826e-02, -6.61107357e-02,
       -3.15538279e-02, -4.46460827e-02, -3.81135647e-02,
       -6.43782753e-03, -1.82115680e-02, -2.03619962e-02,
       -2.23063741e-02, -5.94574259e-02, -3.86550118e-02,
       -3.02092370e-02, -4.73602223e-02, -5.15699908e-02,
       -4.69186781e-02, -2.40946368e-02, -1.46059349e-01,
       -3.28436279e-02, -2.13562845e-02, -3.34699714e-02,
       -7.66677179e-02, -4.32256913e-02, -3.52826435e-02,
       -5.94574259e-02, -1.93166852e-02, -4.95094339e-02,
       -3.58666114e-02, -1.82115680e-02, -2.40946368e-02,
        3.12443417e-01, -4.77977239e-02, -2.32176916e-02],
      [ 2.72704528e-01, -1.18647289e-01,  6.92124563e-01,
       -1.49654480e-01, -2.19627094e-01,  7.52664415e-01,
       -1.68280263e+00, -4.08317376e-01,  2.73899875e+00,
       -2.13562845e-02, -1.66780219e-01,  1.07120139e+00,
       -1.01465417e+00, -3.91866439e-01,  1.70979959e+00,
       -6.77628195e-01, -1.70350088e-02, -2.17406589e-01,
       -3.66626061e-01, -3.92358867e-01,  1.19476558e+00,
       -5.88693059e-01, -1.75000915e-01, -4.17325513e-01,
       -3.44211323e-01, -2.21349813e-01, -9.61445151e-02,
       -1.75000915e-01, -3.22880042e-01, -8.71786495e-02,
        4.04856915e-01, -2.49408357e-02, -5.94574259e-02,
       -4.77977239e-02, -4.27418202e-02, -5.62111112e-02,
       -4.55686097e-02, -2.95140668e-02, -5.83951091e-02,
       -5.35521281e-02, -2.88021826e-02, -6.61107357e-02,
       -3.15538279e-02, -4.46460827e-02, -3.81135647e-02,
       -6.43782753e-03, -1.82115680e-02, -2.03619962e-02,
       -2.23063741e-02, -5.94574259e-02, -3.86550118e-02,
       -3.02092370e-02, -4.73602223e-02, -5.15699908e-02,
       -4.69186781e-02, -2.40946368e-02, -1.46059349e-01,
       -3.28436279e-02, -2.13562845e-02, -3.34699714e-02,
       -7.66677179e-02, -4.32256913e-02, -3.52826435e-02,
       -5.94574259e-02, -1.93166852e-02, -4.95094339e-02,
       -3.58666114e-02, -1.82115680e-02, -2.40946368e-02,
        3.12443417e-01, -4.77977239e-02, -2.32176916e-02],
      [ 5.01169214e-01,  2.46631022e+00, -1.44482663e+00,
       -1.49654480e-01,  4.65776107e+00, -9.08669669e-01,
        5.94246755e-01, -4.08317376e-01, -3.65096917e-01,
       -2.13562845e-02, -1.66780219e-01,  1.07120139e+00,
       -1.01465417e+00,  2.55188988e+00, -5.84863867e-01,
       -6.77628195e-01, -1.70350088e-02, -2.17406589e-01,
       -3.66626061e-01, -3.92358867e-01, -8.36984273e-01,
       -5.88693059e-01, -1.75000915e-01, -4.17325513e-01,
       -3.44211323e-01,  4.51773593e+00, -9.61445151e-02,
       -1.75000915e-01, -3.22880042e-01, -8.71786495e-02,
        4.04856915e-01, -2.49408357e-02, -5.94574259e-02,
       -4.77977239e-02, -4.27418202e-02, -5.62111112e-02,
       -4.55686097e-02, -2.95140668e-02, -5.83951091e-02,
       -5.35521281e-02, -2.88021826e-02, -6.61107357e-02,
       -3.15538279e-02, -4.46460827e-02, -3.81135647e-02,
       -6.43782753e-03, -1.82115680e-02, -2.03619962e-02,
       -2.23063741e-02, -5.94574259e-02, -3.86550118e-02,
       -3.02092370e-02, -4.73602223e-02, -5.15699908e-02,
       -4.69186781e-02, -2.40946368e-02, -1.46059349e-01,
       -3.28436279e-02, -2.13562845e-02, -3.34699714e-02,
       -7.66677179e-02, -4.32256913e-02, -3.52826435e-02,
       -5.94574259e-02, -1.93166852e-02, -4.95094339e-02,
       -3.58666114e-02, -1.82115680e-02, -2.40946368e-02,
        3.12443417e-01, -4.77977239e-02, -2.32176916e-02]])
```

KNN Classifier

In [44]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [45]:
```
k=4
neigh = KNeighborsClassifier(n_neighbors=k).fit(x_train_sc, y_train)
neigh
```

Out[45]:
```
KNeighborsClassifier(n_neighbors=4)
```

In [46]:
```
yhat = neigh.predict(x_test_sc)
yhat[0:5]
```

Out[46]:
```
array([0, 0, 0, 0, 1])
```

In [47]:
```
from sklearn import metrics
```

checking accuracy

In [48]:
```
score = metrics.accuracy_score(y_test, yhat)
print('train set accuracy: ', metrics.accuracy_score(y_train, neigh.predict(x_trai
print('test set accuracy: ', score)
```

```
train set accuracy:  0.8716896680343156
test set accuracy:  0.8112050389524283
```

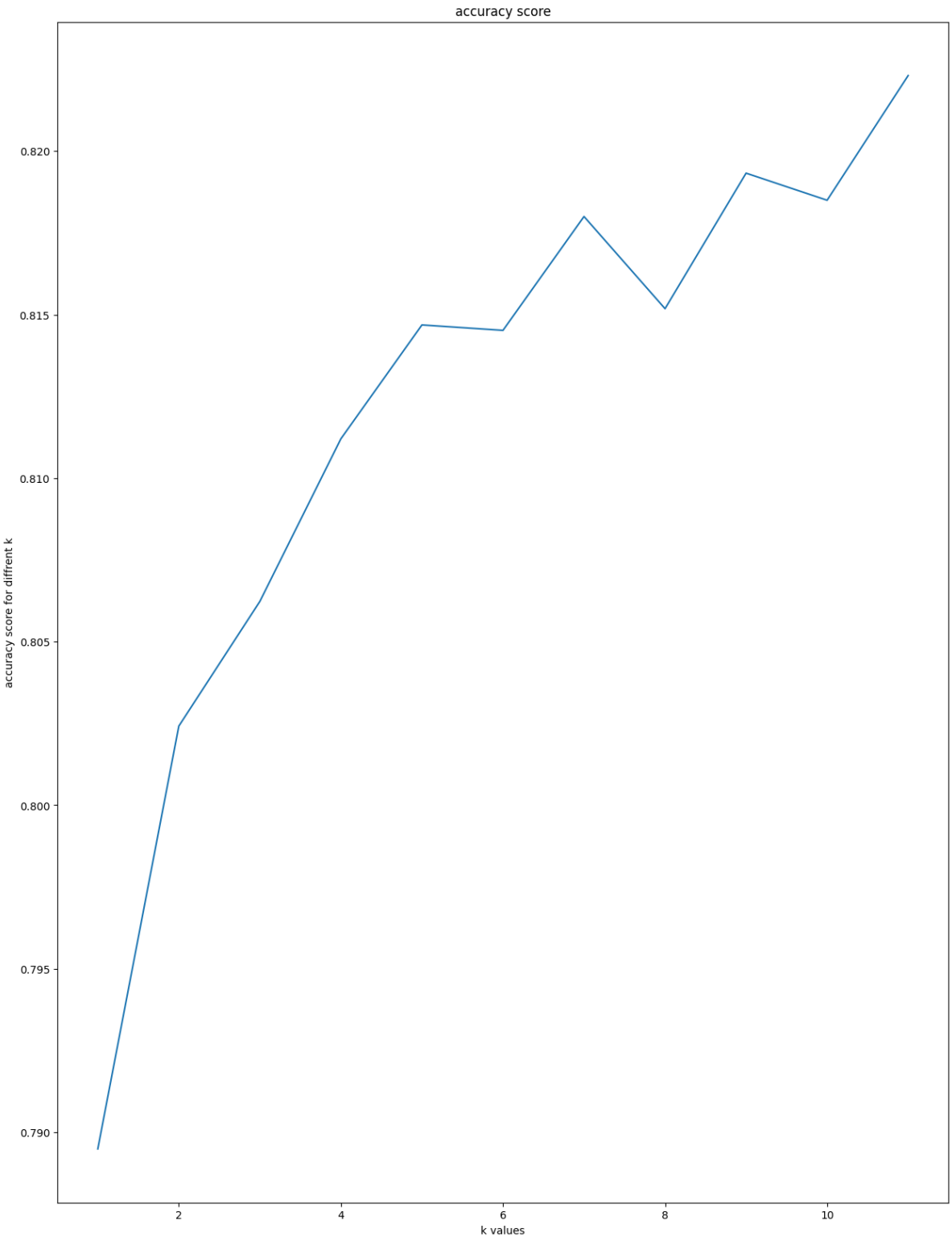
In [49]:
```
ks=12
mean_acc = np.zeros(ks-1)
std_acc = np.zeros(ks-1)
k_values = np.zeros(ks-1)

for n in range(1,ks):
    k_values[n-1] = n
    neigh = KNeighborsClassifier(n_neighbors=n).fit(x_train_sc, y_train)
    yhat = neigh.predict(x_test_sc)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

    std_acc[n-1] = np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

Out[49]:
```
array([0.78949113, 0.80242002, 0.80623239, 0.81120504, 0.81468589,
       0.81452014, 0.81800099, 0.81518316, 0.81932703, 0.81849826,
       0.82231062])
```

In [50]:
```
std_acc
```

Out[50]:
```
array([0.00524858, 0.00512632, 0.00508867, 0.00503841, 0.00500245,
       0.00500418, 0.00496758, 0.00499726, 0.00495346, 0.0049623 ,
       0.00492132])
```

In [51]:
```
plt.plot(k_values, mean_acc)
plt.title('accuracy score')
plt.xlabel('k values')
plt.ylabel('accuracy score for diffrent k')
plt.show()
```



confusion_matrix

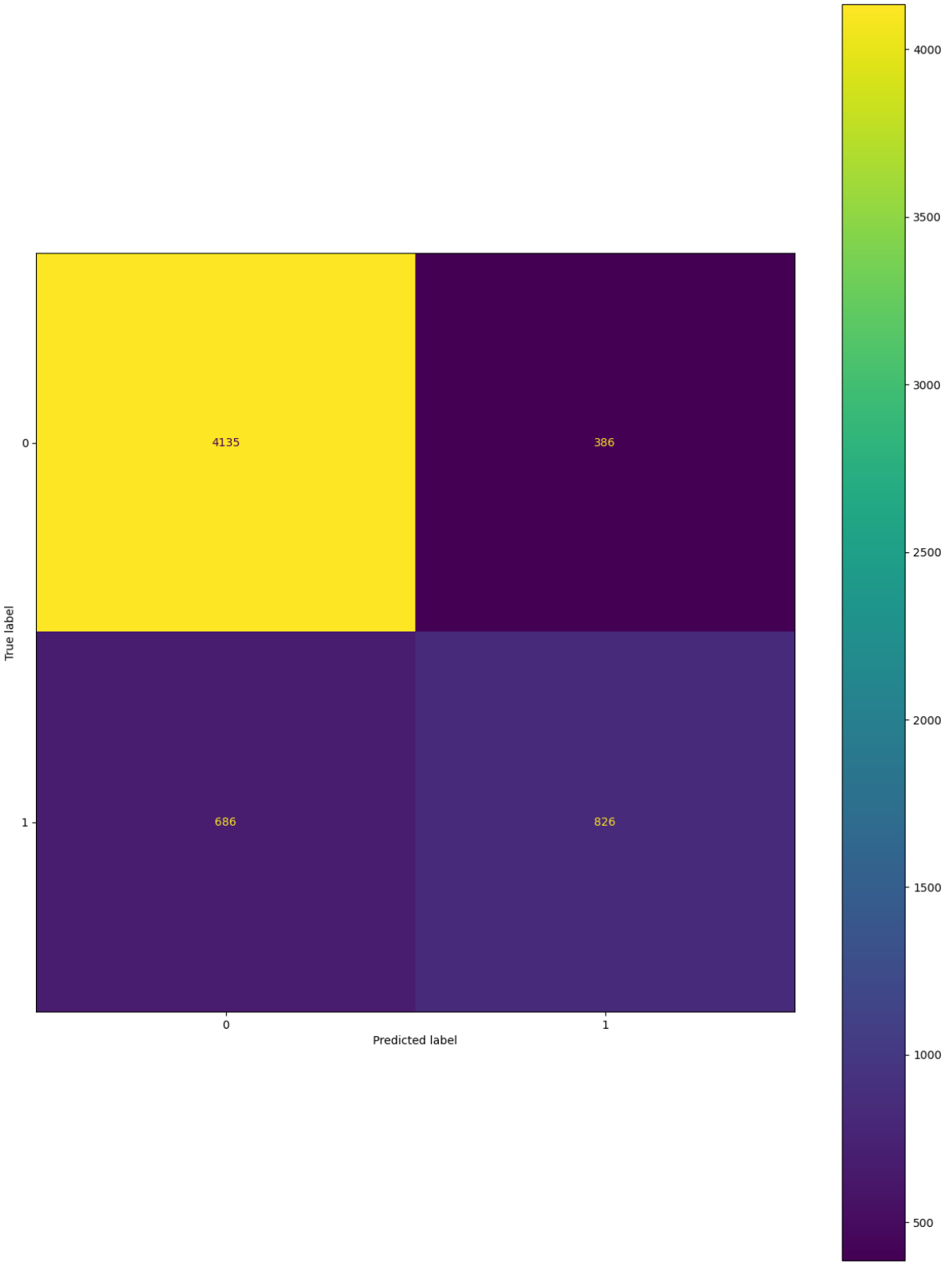
In [52]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

In [53]:
```
cm = confusion_matrix(y_test,yhat)
cm
```

Out[53]:
```
array([[4135,  386],
       [ 686,  826]])
```

In [54]:
```
cm = confusion_matrix(y_test, yhat, labels=neigh.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
```

```
                                display_labels=neigh.classes_)
disp.plot()

plt.show()
```



result is not very good. It seems because of the data is imbalance

resampling

In [55]:
```
from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='minority')
x_sm, y_sm = smote.fit_resample(x_train, y_train)
```

In [56]:
```
x_sm.shape
```

Out[56]:
```
(36266, 72)
```

In [57]:
```
scaler = StandardScaler().fit(x_sm)
x_train_sm_sc = scaler.transform(x_sm)

x_train_sm_sc[0:5]
```

Out[57]:
```
array([[ 6.32962763e-01, -1.54669339e+00,  6.03427256e-01,
        -1.94828930e-01, -2.62490510e-01,  1.34468528e-01,
         6.36885223e-01, -4.25091694e-01, -3.95844713e-01,
        -1.74185656e-02, -1.49429010e-01,  8.23087087e-01,
        -7.86293056e-01, -4.54447101e-01, -6.28656754e-01,
         1.56897912e+00, -1.38944420e-02, -1.87599311e-01,
        -3.72442550e-01, -3.39987043e-01,  9.47649893e-01,
        -5.11676525e-01, -1.47709181e-01, -3.35021520e-01,
        -2.96402116e-01, -2.60043410e-01, -8.83487016e-02,
        -1.77121319e-01, -2.93361754e-01, -7.72085637e-02,
         3.75048347e-01, -2.67850573e-02, -6.37956121e-02,
        -5.33686708e-02, -3.51831915e-02, -5.64012543e-02,
        -3.78837019e-02, -2.75726930e-02, -5.28465271e-02,
        -5.88105071e-02, -3.19574920e-02, -7.42799373e-02,
        -3.23869178e-02, -3.81402855e-02, -3.28024282e-02,
        -5.25117098e-03, -1.48539881e-02, -1.66077216e-02,
        -2.12310075e-02, -6.46613346e-02, -4.00231926e-02,
        -2.64645561e-02, -4.76045691e-02, -4.73127532e-02,
        -4.78946231e-02, -2.09915851e-02, -1.24846695e-01,
        -2.78011585e-02, -1.74185656e-02, -2.72956713e-02,
        -7.81236996e-02, -4.20459050e-02, -3.27324926e-02,
        -5.41425519e-02, -1.76663161e-02, -4.49106291e-02,
        -4.07085236e-02, -1.57552508e-02, -2.17664984e-02,
         2.97160785e-01, -4.18424779e-02, -2.27260742e-02],
       [ 1.36622394e+00,  2.25181325e+00,  6.03427256e-01,
        -1.94828930e-01,  3.86454760e+00, -2.14813407e-01,
         6.36885223e-01, -4.25091694e-01, -3.95844713e-01,
        -1.74185656e-02, -1.49429010e-01,  8.23087087e-01,
        -7.86293056e-01, -4.54447101e-01, -6.28656754e-01,
         1.56897912e+00, -1.38944420e-02, -1.87599311e-01,
        -3.72442550e-01, -3.39987043e-01,  9.47649893e-01,
        -5.11676525e-01, -1.47709181e-01, -3.35021520e-01,
        -2.96402116e-01, -2.60043410e-01, -8.83487016e-02,
        -1.77121319e-01, -2.93361754e-01, -7.72085637e-02,
         3.75048347e-01, -2.67850573e-02, -6.37956121e-02,
        -5.33686708e-02, -3.51831915e-02, -5.64012543e-02,
        -3.78837019e-02, -2.75726930e-02, -5.28465271e-02,
        -5.88105071e-02, -3.19574920e-02,  1.34625854e+01,
        -3.23869178e-02, -3.81402855e-02, -3.28024282e-02,
        -5.25117098e-03, -1.48539881e-02, -1.66077216e-02,
        -2.12310075e-02, -6.46613346e-02, -4.00231926e-02,
        -2.64645561e-02, -4.76045691e-02, -4.73127532e-02,
        -4.78946231e-02, -2.09915851e-02, -1.24846695e-01,
        -2.78011585e-02, -1.74185656e-02, -2.72956713e-02,
        -7.81236996e-02, -4.20459050e-02, -3.27324926e-02,
        -5.41425519e-02, -1.76663161e-02, -4.49106291e-02,
        -4.07085236e-02, -1.57552508e-02, -2.17664984e-02,
        -3.37279271e+00, -4.18424779e-02, -2.27260742e-02],
       [-7.52086136e-01, -9.13608949e-01,  6.03427256e-01,
        -1.94828930e-01, -2.62490510e-01, -2.14813407e-01,
         6.36885223e-01, -4.25091694e-01, -3.95844713e-01,
        -1.74185656e-02, -1.49429010e-01,  8.23087087e-01,
        -7.86293056e-01, -4.54447101e-01, -6.28656754e-01,
         1.56897912e+00, -1.38944420e-02, -1.87599311e-01,
        -3.72442550e-01, -3.39987043e-01,  9.47649893e-01,
        -5.11676525e-01, -1.47709181e-01, -3.35021520e-01,
        -2.96402116e-01, -2.60043410e-01, -8.83487016e-02,
        -1.77121319e-01, -2.93361754e-01, -7.72085637e-02,
         3.75048347e-01, -2.67850573e-02, -6.37956121e-02,
        -5.33686708e-02, -3.51831915e-02, -5.64012543e-02,
        -3.78837019e-02, -2.75726930e-02, -5.28465271e-02,
```

```
       -5.88105071e-02, -3.19574920e-02, -7.42799373e-02,
       -3.23869178e-02, -3.81402855e-02, -3.28024282e-02,
       -5.25117098e-03, -1.48539881e-02, -1.66077216e-02,
       -2.12310075e-02, -6.46613346e-02, -4.00231926e-02,
       -2.64645561e-02, -4.76045691e-02, -4.73127532e-02,
       -4.78946231e-02, -2.09915851e-02, -1.24846695e-01,
       -2.78011585e-02, -1.74185656e-02, -2.72956713e-02,
       -7.81236996e-02, -4.20459050e-02, -3.27324926e-02,
       -5.41425519e-02, -1.76663161e-02, -4.49106291e-02,
       -4.07085236e-02, -1.57552508e-02, -2.17664984e-02,
        2.97160785e-01, -4.18424779e-02, -2.27260742e-02],
      [ 1.44121975e-01, -2.80524509e-01,  6.03427256e-01,
       -1.94828930e-01, -2.62490510e-01,  6.58391430e-01,
       -1.58421211e+00, -4.25091694e-01,  2.54798027e+00,
       -1.74185656e-02, -1.49429010e-01,  8.23087087e-01,
       -7.86293056e-01, -4.54447101e-01,  1.60471473e+00,
       -6.42491987e-01, -1.38944420e-02, -1.87599311e-01,
       -3.72442550e-01, -3.39987043e-01,  9.47649893e-01,
       -5.11676525e-01, -1.47709181e-01, -3.35021520e-01,
       -2.96402116e-01, -2.60043410e-01, -8.83487016e-02,
       -1.77121319e-01, -2.93361754e-01, -7.72085637e-02,
        3.75048347e-01, -2.67850573e-02, -6.37956121e-02,
       -5.33686708e-02, -3.51831915e-02, -5.64012543e-02,
       -3.78837019e-02, -2.75726930e-02, -5.28465271e-02,
       -5.88105071e-02, -3.19574920e-02, -7.42799373e-02,
       -3.23869178e-02, -3.81402855e-02, -3.28024282e-02,
       -5.25117098e-03, -1.48539881e-02, -1.66077216e-02,
       -2.12310075e-02, -6.46613346e-02, -4.00231926e-02,
       -2.64645561e-02, -4.76045691e-02, -4.73127532e-02,
       -4.78946231e-02, -2.09915851e-02, -1.24846695e-01,
       -2.78011585e-02, -1.74185656e-02, -2.72956713e-02,
       -7.81236996e-02, -4.20459050e-02, -3.27324926e-02,
       -5.41425519e-02, -1.76663161e-02, -4.49106291e-02,
       -4.07085236e-02, -1.57552508e-02, -2.17664984e-02,
        2.97160785e-01, -4.18424779e-02, -2.27260742e-02],
      [ 3.88542369e-01,  2.25181325e+00, -1.66968713e+00,
       -1.94828930e-01,  3.86454760e+00, -1.08801824e+00,
        6.36885223e-01, -4.25091694e-01, -3.95844713e-01,
       -1.74185656e-02, -1.49429010e-01,  8.23087087e-01,
       -7.86293056e-01,  2.22281513e+00, -6.28656754e-01,
       -6.42491987e-01, -1.38944420e-02, -1.87599311e-01,
       -3.72442550e-01, -3.39987043e-01, -1.05920040e+00,
       -5.11676525e-01, -1.47709181e-01, -3.35021520e-01,
       -2.96402116e-01,  3.86224303e+00, -8.83487016e-02,
       -1.77121319e-01, -2.93361754e-01, -7.72085637e-02,
        3.75048347e-01, -2.67850573e-02, -6.37956121e-02,
       -5.33686708e-02, -3.51831915e-02, -5.64012543e-02,
       -3.78837019e-02, -2.75726930e-02, -5.28465271e-02,
       -5.88105071e-02, -3.19574920e-02, -7.42799373e-02,
       -3.23869178e-02, -3.81402855e-02, -3.28024282e-02,
       -5.25117098e-03, -1.48539881e-02, -1.66077216e-02,
       -2.12310075e-02, -6.46613346e-02, -4.00231926e-02,
       -2.64645561e-02, -4.76045691e-02, -4.73127532e-02,
       -4.78946231e-02, -2.09915851e-02, -1.24846695e-01,
       -2.78011585e-02, -1.74185656e-02, -2.72956713e-02,
       -7.81236996e-02, -4.20459050e-02, -3.27324926e-02,
       -5.41425519e-02, -1.76663161e-02, -4.49106291e-02,
       -4.07085236e-02, -1.57552508e-02, -2.17664984e-02,
        2.97160785e-01, -4.18424779e-02, -2.27260742e-02]])
```

In [58]:
```
#from sklearn.model_selection import train_test_split
#x_train, x_test, y_train, y_test = train_test_split(x_sm, y_sm, test_size=0.2, rar
#print('train set : ', x_train.shape, y_train.shape)
#print('test set : ', x_test.shape, y_test.shape)
```

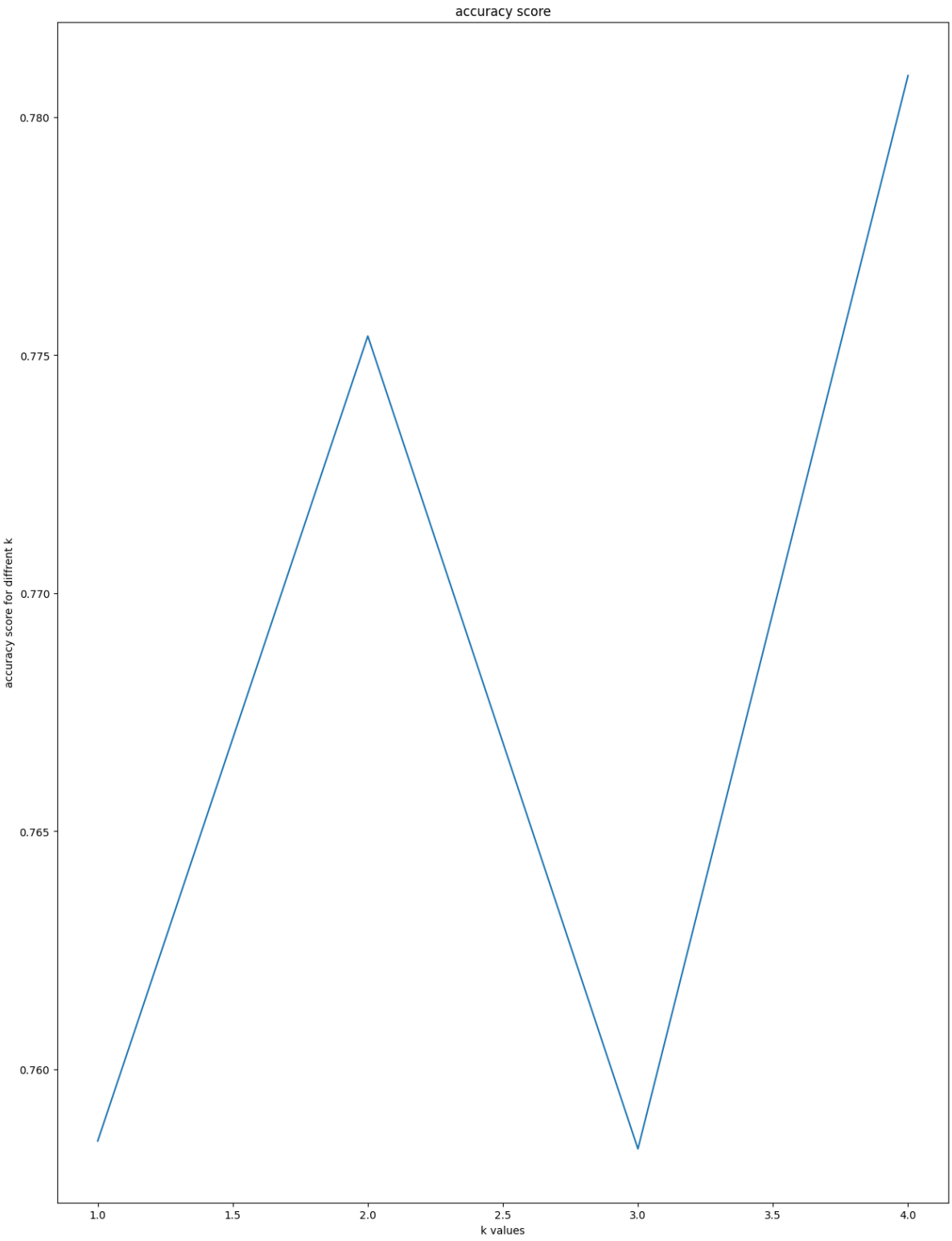
In [59]:
```
ks=5
mean_acc = np.zeros(ks-1)
std_acc = np.zeros(ks-1)
k_values = np.zeros(ks-1)

for n in range(1,ks):
    k_values[n-1] = n
    neigh = KNeighborsClassifier(n_neighbors=n).fit(x_train_sm_sc, y_sm)
    yhat = neigh.predict(x_test_sc)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)

    std_acc[n-1] = np.std(yhat==y_test)/np.sqrt(yhat.shape[0])

mean_acc
```

Out[59]:
```
array([0.75849494, 0.77540196, 0.75832919, 0.78087187])
```

In [60]:
```
std_acc
```

Out[60]:
```
array([0.00551027, 0.00537279, 0.00551156, 0.00532565])
```

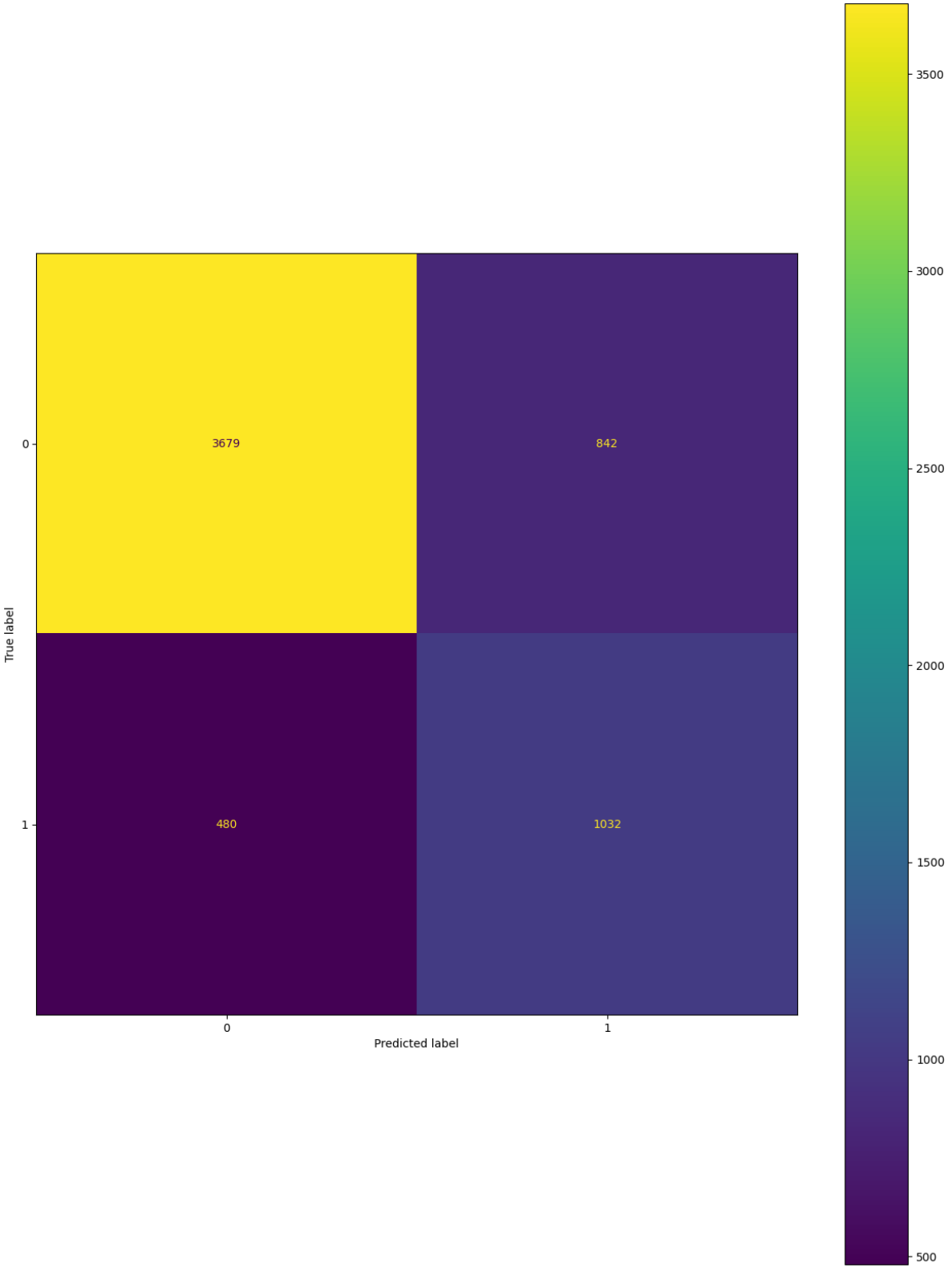
In [61]:
```
plt.plot(k_values, mean_acc)
plt.title('accuracy score')
plt.xlabel('k values')
plt.ylabel('accuracy score for diffrent k')
plt.show()
```



In [62]:
```
cm = confusion_matrix(y_test,yhat)
cm
```

Out[62]:
```
array([[3679,  842],
       [ 480, 1032]])
```

In [63]:
```
cm = confusion_matrix(y_test, yhat, labels=neigh.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=neigh.classes_)
disp.plot()

plt.show()
```



In [64]:
```
from sklearn.decomposition import PCA
```

In [65]:
```
pca = PCA()
X = pca.fit_transform(X)
```

In [66]:
```
explained_variance = pca.explained_variance_ratio_
explained_variance
```

Out[66]:
```
array([5.79571725e-02, 3.52399993e-02, 3.24968329e-02, 2.52870074e-02,
       2.31687439e-02, 2.25710701e-02, 2.11493986e-02, 1.92309565e-02,
       1.79203093e-02, 1.74537062e-02, 1.66380593e-02, 1.61236900e-02,
       1.57505097e-02, 1.52322815e-02, 1.48895960e-02, 1.47365892e-02,
       1.45378297e-02, 1.41957764e-02, 1.41141798e-02, 1.40788879e-02,
       1.39982388e-02, 1.39807920e-02, 1.39499578e-02, 1.39401659e-02,
       1.39338989e-02, 1.39320767e-02, 1.39219802e-02, 1.39193668e-02,
       1.39166098e-02, 1.39136552e-02, 1.39122280e-02, 1.39099308e-02,
       1.39086435e-02, 1.39072973e-02, 1.39049212e-02, 1.39042376e-02,
       1.39025376e-02, 1.39018555e-02, 1.39002515e-02, 1.38974272e-02,
       1.38961068e-02, 1.38921381e-02, 1.38878212e-02, 1.38748057e-02,
       1.38576761e-02, 1.38371452e-02, 1.38155953e-02, 1.37930986e-02,
       1.37463666e-02, 1.37307440e-02, 1.36973168e-02, 1.36259684e-02,
       1.34968324e-02, 1.33274044e-02, 1.28964681e-02, 1.27336244e-02,
       1.23985109e-02, 1.21155495e-02, 1.15019359e-02, 1.05350611e-02,
       1.01595341e-02, 7.77980958e-03, 7.21060589e-03, 5.27889897e-03,
       3.38188297e-03, 3.00430647e-04, 4.03323484e-31, 2.79470471e-31,
       5.14878697e-32, 4.79420908e-32, 4.01140094e-32, 2.01631962e-32])
```

In [67]:
```
pca = PCA(n_components=7)
x_train_pca = pca.fit_transform(x_sm)
x_test_pca = pca.transform(x_test)
x_train_pca[0:5]
```

Out[67]:
```
array([[ 1.43169768, -0.59097965, -2.23928972, -0.0139588 ,  0.10563559,
        -0.72490462,  0.08336794],
       [ 2.41599741,  2.49651627, -1.38789926, -1.14411502,  0.15726729,
         5.33329392,  4.52697525],
       [ 1.17308963, -0.56946363, -2.17694713, -0.03964841,  0.19770079,
        -0.40739446, -0.14309468],
       [ 2.23354063, -0.5340131 ,  0.68346279, -0.29049687, -0.92466716,
        -0.73186298,  0.98450889],
       [ 0.58831779,  0.20894241,  2.89002152, -2.71907578,  3.72990137,
         2.27709668, -0.72377956]])
```

In [68]:
```
scaler = StandardScaler().fit(x_train_pca)
x_train_pca_sc = scaler.transform(x_train_pca)

x_test_pca_sc = scaler.transform(x_test_pca)
x_train_pca_sc[0:5]
```

Out[68]:
```
array([[ 0.71185423, -0.37215492, -1.44428711, -0.00990895,  0.07803398,
        -0.5625604 ,  0.06643613],
       [ 1.20125779,  1.57211981, -0.89516108, -0.81217405,  0.11617479,
         4.13888926,  3.60755865],
       [ 0.58327176, -0.35860574, -1.40407766, -0.02814526,  0.14604339,
        -0.31615744, -0.11403254],
       [ 1.11053848, -0.33628163,  0.44081679, -0.2062153 , -0.6830601 ,
        -0.56796042,  0.78455776],
       [ 0.29251742,  0.13157635,  1.8639932 , -1.93019298,  2.7553123 ,
         1.76713512, -0.57678186]])
```

In [69]:
```
k=4
neigh_pca = KNeighborsClassifier(n_neighbors=k).fit(x_train_pca_sc, y_sm)
neigh_pca
```

Out[69]:
```
KNeighborsClassifier(n_neighbors=4)
```

In [70]:
```
yhat_pca = neigh_pca.predict(x_test_pca_sc)
yhat_pca[0:5]
```

Out[70]:
```
array([0, 0, 0, 0, 1])
```

In [71]:
```
y_test.shape
```

Out[71]:
```
(6033,)
```

In [72]:
```
yhat_pca.shape
```

Out[72]:
```
(6033,)
```

In [73]:
```
score = metrics.accuracy_score(y_test, yhat_pca)
print('train set accuracy: ', metrics.accuracy_score(y_sm, neigh_pca.predict(x_trai
print('test set accuracy: ', score)
```

```
train set accuracy:  0.8973970109744664
test set accuracy:  0.7961213326703133
```

In [74]:
```
cm = confusion_matrix(y_test,yhat_pca)
cm
```

Out[74]:
```
array([[3811,  710],
       [ 520,  992]])
```