In [27]:
```
import numpy as np
```

In [28]:
```
import pandas as pd
```

In [29]:
```
import matplotlib.pyplot as plt
```

In [30]:
```
import seaborn as sns
```

In [31]:
```
%matplotlib inline
```

In [32]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [33]:
```
df=pd.read_csv(r"D:\Data Science with AI\Data Science With AI\3rd, 4th -august-
```

In [34]:
```
df.shape
```

Out[34]:
```
(32561, 15)
```

In [35]:
```
df.head()
```

Out[35]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relati
0	90	?	77053	HS-grad	9	Widowed	?	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	
2	66	?	186061	Some-college	10	Widowed	?	Unr
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unr
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Ow

In [36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [37]:
```
df[df=='?']=np.nan
```

In [38]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       30725 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      30718 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  31978 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [39]:
```
for col in ['workclass','occupation','native.country']:
    df[col].fillna(df[col].mode()[0],inplace=True)
```

In [40]:
```
df.head()
```

Out[40]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relati
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty	Unr
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unr
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Ow

In [41]:
```
df.isnull().sum()
```

Out[41]:
```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

In [42]:
```
x=df.drop(['income'],axis=1)
y=df['income']
```

In [43]:
```
x
```

Out[43]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct
4	41	Private	264663	Some-college	10	Separated	Prof-specialty
...	...	...	...	...	...	...	...
32556	22	Private	310152	Some-college	10	Never-married	Protective-serv
32557	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support
32558	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct
32559	58	Private	151910	HS-grad	9	Widowed	Adm-clerical
32560	22	Private	201490	HS-grad	9	Never-married	Adm-clerical

32561 rows × 14 columns

In [44]:
```
y
```

Out[44]:
```
0        <=50K
1        <=50K
2        <=50K
3        <=50K
4        <=50K
         ...
32556    <=50K
32557    <=50K
32558     >50K
32559    <=50K
32560    <=50K
Name: income, Length: 32561, dtype: object
```

In [45]:
```
x.head()
```

Out[45]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relati
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty	Unr
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unr
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Ow

In [46]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [47]:
```
from sklearn import preprocessing
categorical=['workclass','education','marital.status','occupation','relationship
for feature in categorical:
    le=preprocessing.LabelEncoder()
    x_train[feature]=le.fit_transform(x_train[feature])
    x_test[feature]=le.transform(x_test[feature])
```

In [48]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=pd.DataFrame(scaler.fit_transform(x_train),columns=x.columns)
x_test=pd.DataFrame(scaler.transform(x_test),columns=x.columns)
```

In [49]:
```
x_train.head()
```

Out[49]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupatio
0	0.101484	2.600478	-1.494279	-0.332263	1.133894	-0.402341	-0.78223
1	0.028248	-1.884720	0.438778	0.184396	-0.423425	-0.402341	-0.02669
2	0.247956	-0.090641	0.045292	1.217715	-0.034095	0.926666	-0.78223
3	-0.850587	-1.884720	0.793152	0.184396	-0.423425	0.926666	-0.53038
4	-0.044989	-2.781760	-0.853275	0.442726	1.523223	-0.402341	-0.78223

Logistic Regression model with all features

In [50]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
```

```
y_pred=logreg.predict(x_test)
print('Logistic Regression accuracy score with all the features:{0:0.4f}'.format
```

```
Logistic Regression accuracy score with all the features:0.8218
```

Logistic regression with PCA

In [25]:
```
from sklearn.decomposition import PCA
pca=PCA()
x_train=pca.fit_transform(x_train)
pca.explained_variance_ratio_
```

Out[25]:
```
array([0.14757168, 0.10182915, 0.08147199, 0.07880174, 0.07463545,
       0.07274281, 0.07009602, 0.06750902, 0.0647268 , 0.06131155,
       0.06084207, 0.04839584, 0.04265038, 0.02741548])
```

In [26]:
```
x.head()
```

Out[26]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relati
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty	Unr
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unr
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Ow

Logistic Regression with first 13 features

In [52]:
```
x=df.drop(['income','native.country'],axis=1)
y=df['income']


x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
categorical=['workclass','education','marital.status','occupation','relationship
for feature in categorical:
    le=preprocessing.LabelEncoder()
    x_train[feature]=le.fit_transform(x_train[feature])
    x_test[feature]=le.transform(x_test[feature])
x_train=pd.DataFrame(scaler.fit_transform(x_train),columns=x.columns)
x_test=pd.DataFrame(scaler.transform(x_test),columns=x.columns)
logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
print('Logistic Regression accuracy score with the first 13 features:{0:0.4f}'.f
```

```
Logistic Regression accuracy score with the first 13 features:0.8213
```

Logistic Regression with first 12 features

In [54]:
```
x=df.drop(['income','native.country','hours.per.week'],axis=1)
y=df['income']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
categorical=['workclass','education','marital.status','occupation','relationship
for feature in categorical:
    le=preprocessing.LabelEncoder()
    x_train[feature]=le.fit_transform(x_train[feature])
    x_test[feature]=le.transform(x_test[feature])

x_train=pd.DataFrame(scaler.fit_transform(x_train),columns=x.columns)
x_test=pd.DataFrame(scaler.transform(x_test),columns=x.columns)

logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)
print('Logistic Regression accuracy score with the first 12 features:{0:0.4f}'.f
```

```
Logistic Regression accuracy score with the first 12 features:0.8227
```

Logistic Regression with first 11 features

In [56]:
```
x=df.drop(['income','native.country','hours.per.week','capital.loss'],axis=1)
y=df['income']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)


categorical=['workclass','education','marital.status','occupation','relationship
for feature in categorical:
    le=preprocessing.LabelEncoder()
    x_train[feature]=le.fit_transform(x_train[feature])
    x_test[feature]=le.transform(x_test[feature])

x_train=pd.DataFrame(scaler.fit_transform(x_train),columns=x.columns)
x_test=pd.DataFrame(scaler.transform(x_test),columns=x.columns)


logreg=LogisticRegression()
logreg.fit(x_train,y_train)
y_pred=logreg.predict(x_test)

print('Logistic Regression accuracy score with the first 11 features:{0:0.4f}'.f
```

```
Logistic Regression accuracy score with the first 11 features:0.8186
```

select right number of dimensions

In [57]:
```
x=df.drop(['income'],axis=1)
y=df['income']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

```
categorical=['workclass','education','marital.status','occupation','relationship
for feature in categorical:
    le=preprocessing.LabelEncoder()
    x_train[feature]=le.fit_transform(x_train[feature])
    x_test[feature]=le.transform(x_test[feature])

x_train=pd.DataFrame(scaler.fit_transform(x_train),columns=x.columns)

pca=PCA()
pca.fit(x_train)
cumsum=np.cumsum(pca.explained_variance_ratio_)
dim=np.argmax(cumsum>=0.90)+1
print('The number of dimensions required to preserve 90% of variance is',dim)
```

```
The number of dimensions required to preserve 90% of variance is 12
```

In [ ]: