In [111]:

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

In [133]:

```
#Importing the Dataset
data=pd.read_csv('Dataset.csv')
print(data)
print(data.columns)
```

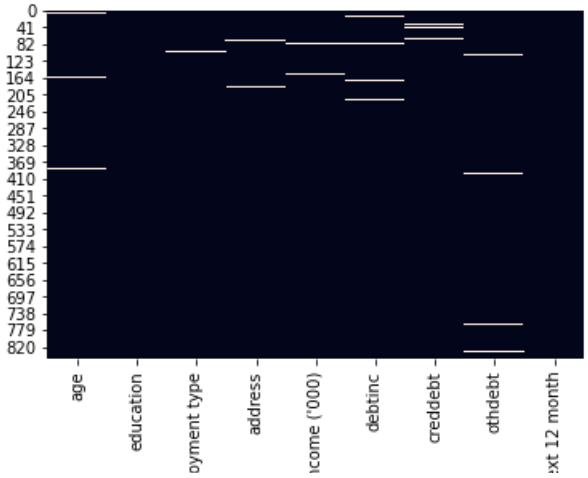
```
      age  education  employment type  address  income ('000)  debtinc  \
0    41.0          3             17.0     12.0          176.0      9.3
1    27.0          1             10.0      6.0           31.0     17.3
2    40.0          1             15.0     14.0           55.0      5.5
3    41.0          1             15.0     14.0          120.0      2.9
4    24.0          2              2.0      0.0           28.0     17.3
..    ...        ...              ...      ...            ...      ...
845  34.0          1             12.0     15.0           32.0      2.7
846  32.0          2             12.0     11.0          116.0      5.7
847  48.0          1             13.0     11.0           38.0     10.8
848  35.0          2              1.0     11.0           24.0      7.8
849  37.0          1             20.0     13.0           41.0     12.9

      creddebt   othdebt  default_next 12 month
0    11.359392  5.008608                      1
1     1.362202  4.000798                      0
2     0.856075  2.168925                      0
3     2.658720  0.821280                      0
4     1.787436  3.056564                      1
..         ...       ...                    ...
845   0.239328  0.624672                      0
846   4.026708  2.585292                      1
847   0.722304  3.381696                      0
848   0.417456  1.454544                      1
849   0.899130  4.389870                      0

[850 rows x 9 columns]
Index(['age', 'education', 'employment type', 'address', 'income ('000)',
       'debtinc', 'creddebt', 'othdebt', 'default_next 12 month'],
      dtype='object')
```

In [134]:

```
sns.heatmap(data.isnull(), cbar=False)
```

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x244dcfcb5e0>
```



empl in default_ne

In [135]:

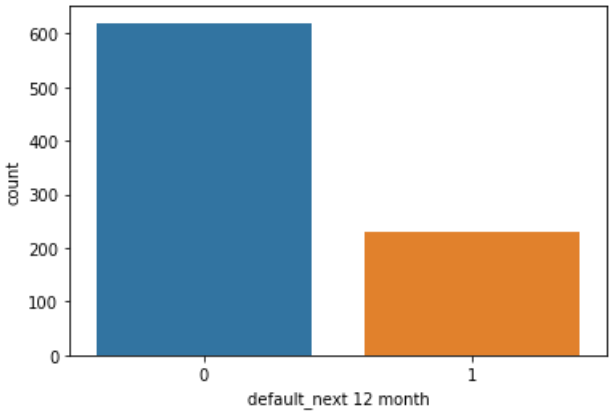
```
#Checking for missing value in data
print(np.sum(data.isna())) #represent total missing observation in each column
print(data.shape)
```

```
age                        9
education                  0
employment type            5
address                    5
income ('000)              5
debtinc                    6
creddebt                  15
othdebt                   11
default_next 12 month      0
dtype: int64
(850, 9)
```

In [136]:

```
y_old= data.loc[:,'default_next 12 month']
X_old= data.loc[:,['age', 'education', 'employment type', 'address',"income ('000)",'debtinc', 'cre
ddebt', 'othdebt']]
sns.countplot(y_old)
y_old.value_counts()
print('Default rate = ',y_old.mean())
```

```
Default rate =  0.27058823529411763
```

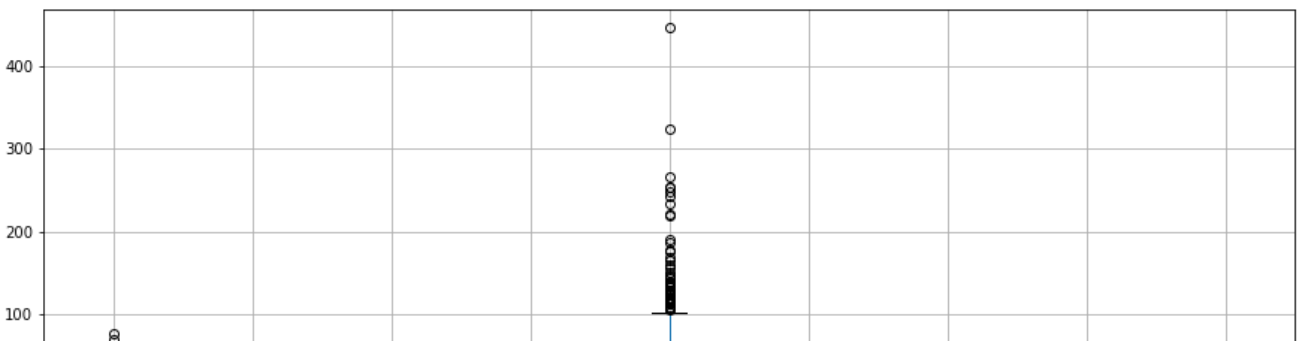


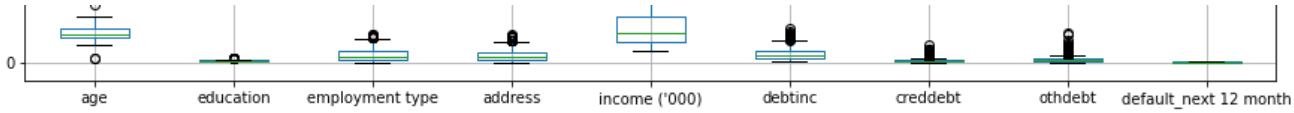
In [137]:

```
#Checking outliers using  boxplot
data.boxplot(figsize= (15,5))
```

Out[137]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x244dcea9a00>
```





In [138]:

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=2, weights="uniform")
x = imputer.fit_transform(X_old)
```

In [139]:

```
X_imp=pd.DataFrame(x)
X_imp.columns = ['age', 'education', 'employment type', 'address',"income ('000)",'debtinc', 'credd
ebt', 'othdebt']
np.sum(X_imp.isna())
```

Out[139]:

```
age                 0
education           0
employment type     0
address             0
income ('000)       0
debtinc             0
creddebt            0
othdebt             0
dtype: int64
```

In [140]:

```
X_num = X_imp.loc[:,['age',"income ('000)",'debtinc','creddebt','othdebt']]
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X_num)
X_scaled = scaler.transform(X_num)
print(X_scaled.mean(axis=0))
print(X_scaled.std(axis=0))
```

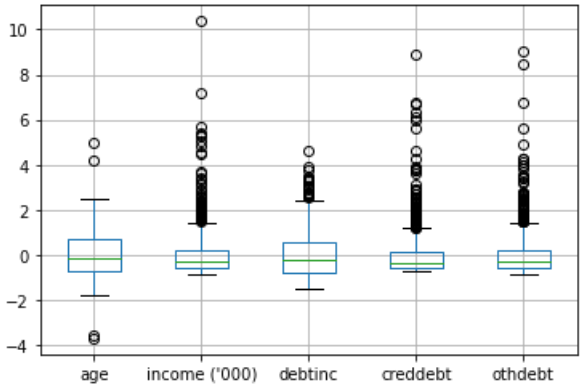
```
[ 3.59451031e-16 -8.35932630e-18  2.08983158e-18 -3.76169684e-17
 -1.67186526e-17]
[1. 1. 1. 1. 1.]
```

In [141]:

```
X_scaled  = pd.DataFrame(X_scaled)
X_scaled.columns = ['age',"income ('000)",'debtinc','creddebt','othdebt']
X_scaled.boxplot()
```

Out[141]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x244de722190>
```



In [142]:

```
X_new= pd.concat( [  X_imp.loc[:,['education', 'employment type', 'address']], X_scaled ],axis=1)
X_new
```

Out[142]:

	education	employment type	address	age	income ('000)	debtinc	creddebt	othdebt
0	3.0	17.0	12.0	0.706747	3.357518	-0.127808	4.594530	0.535635
1	1.0	10.0	6.0	-0.960638	-0.408674	1.063552	-0.102599	0.251296
2	1.0	15.0	14.0	0.587648	0.214696	-0.693705	-0.340401	-0.265541
3	1.0	15.0	14.0	0.706747	1.902989	-1.080897	0.506563	-0.645759
4	2.0	2.0	0.0	-1.317935	-0.486595	1.063552	0.097195	-0.015106
...	...	...	...	...	...	...	...	...
845	1.0	12.0	15.0	-0.126945	-0.382700	-1.110681	-0.630176	-0.701230
846	2.0	12.0	11.0	-0.365143	1.799094	-0.663921	1.149305	-0.148069
847	1.0	13.0	11.0	1.540440	-0.226858	0.095572	-0.403252	0.076625
848	2.0	1.0	11.0	-0.007847	-0.590490	-0.351189	-0.546484	-0.467093
849	1.0	20.0	13.0	0.230351	-0.148936	0.408304	-0.320171	0.361067

850 rows × 8 columns

In [143]:

```
print(X_new.shape)
print(y_old.shape)
```

```
(850, 8)
(850,)
```

In [144]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_new,y_old, test_size=0.20, random_state = 0 )
```

In [145]:

```
X_train.shape
```

Out[145]:

```
(680, 8)
```

In [146]:

```
def mahalanobis(x=None, data=None, cov=None):

    x_mu = x - np.mean(data)
    if not cov:
        cov = np.cov(data.values.T)
    inv_covmat = np.linalg.inv(cov)
    left = np.dot(x_mu, inv_covmat)
    mahal = np.dot(left, x_mu.T)
    return mahal.diagonal()
```

In [147]:

```
#create new column in dataframe that contains Mahalanobis distance for each row
X_train['mahalanobis'] = mahalanobis(x=X_train , data =X_train)
print(X_train.head())
print(y_train.head())
```

```
     education  employment type  address       age  income ('000)    debtinc  \
272        3.0              6.0      4.0 -0.960638      -0.538543  0.036004
372        1.0              8.0     11.0  0.944945      -0.045042 -1.021329
```

```
231         1.0              3.0     11.0  2.374133      -0.174910 -0.321405
10          1.0              0.0      1.0 -0.960638      -0.798280 -1.259601
834         1.0             10.0      1.0 -0.841539      -0.122963 -0.232053

     creddebt   othdebt  mahalanobis
272 -0.305585 -0.377013     3.728877
372 -0.424463 -0.649551     3.629675
231 -0.478006 -0.133537    14.976923
10  -0.656871 -0.852224     6.305129
834 -0.250470 -0.496839     2.519348
272    0
372    0
231    1
10     0
834    0
Name: default_next 12 month, dtype: int64
```

```
<ipython-input-147-e6928c87ed3a>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['mahalanobis'] = mahalanobis(x=X_train , data =X_train)
```

In [148]:

```
from scipy.stats import chi2

#calculate p-value for each mahalanobis distance
X_train['p'] = 1 - chi2.cdf(X_train['mahalanobis'], 7)

#display p-values for first five rows in dataframe
X_train.head()
```

```
<ipython-input-148-c6b5c8ffda3d>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['p'] = 1 - chi2.cdf(X_train['mahalanobis'], 7)
```

Out[148]:

	education	employment type	address	age	income ('000)	debtinc	creddebt	othdebt	mahalanobis	p
272	3.0	6.0	4.0	-0.960638	-0.538543	0.036004	-0.305585	-0.377013	3.728877	0.810422
372	1.0	8.0	11.0	0.944945	-0.045042	-1.021329	-0.424463	-0.649551	3.629675	0.821306
231	1.0	3.0	11.0	2.374133	-0.174910	-0.321405	-0.478006	-0.133537	14.976923	0.036296
10	1.0	0.0	1.0	-0.960638	-0.798280	-1.259601	-0.656871	-0.852224	6.305129	0.504607
834	1.0	10.0	1.0	-0.841539	-0.122963	-0.232053	-0.250470	-0.496839	2.519348	0.925633

In [149]:

```
Xt = X_train[X_train['p']>0.01]
X = Xt.loc[:,['education','employment type','address','age',"income ('000)",'debtinc','creddebt','c
thdebt']]
y = y_train[X_train['p']>0.01]
print(X.shape)
print(y.shape)
```

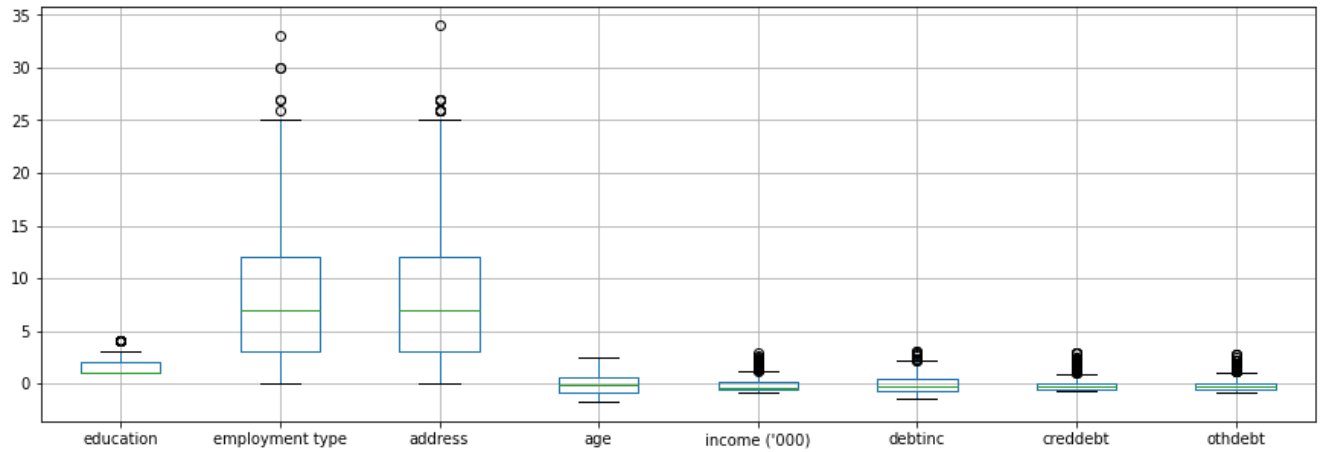
```
(635, 8)
(635,)
```

In [150]:

```
X.boxplot(figsize=(15,5))
```

Out[150]:

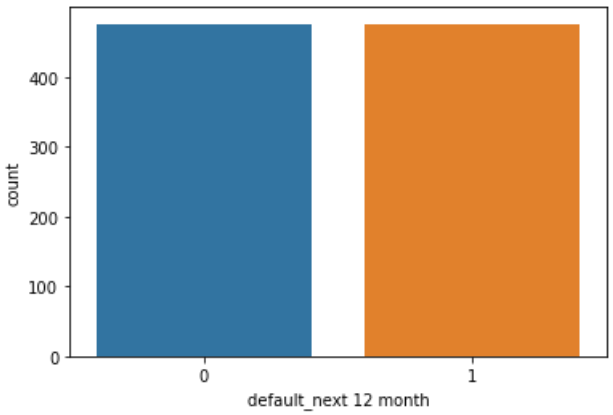
```
<matplotlib.axes._subplots.AxesSubplot at 0x244df858e20>
```



In [151]:

```
from imblearn.over_sampling import RandomOverSampler
os = RandomOverSampler()
X_train_res , y_train_res = os.fit_sample(X ,y)
sns.countplot(y_train_res)
print(X_train_res.shape)
print(y_train_res.value_counts())
```

```
(952, 8)
1    476
0    476
Name: default_next 12 month, dtype: int64
```



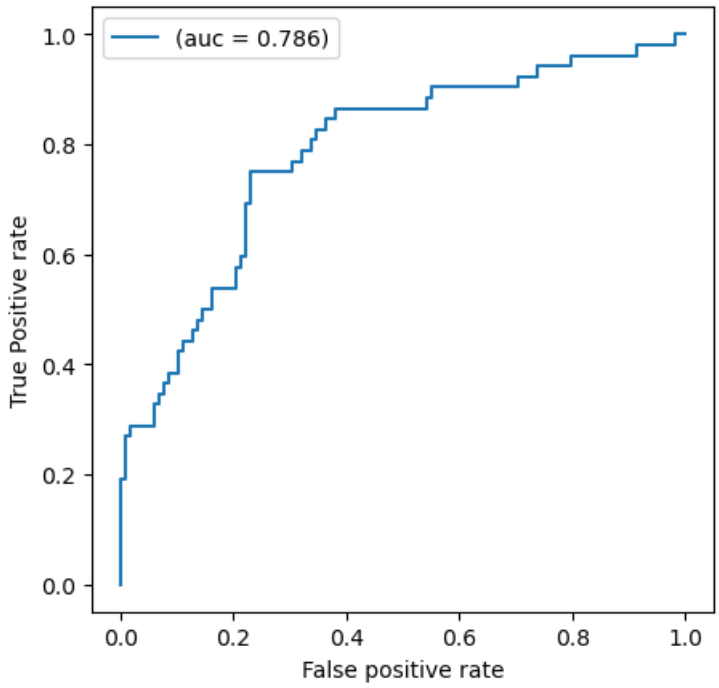
In [152]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 0)
model.fit(X_train_res,y_train_res)
from sklearn.metrics import accuracy_score ,confusion_matrix ,f1_score
y_pred = model.predict(X_train_res)
y_pred_test = model.predict(X_test)
def generate_model_report(y_actual, y_predicted):
    print("Accuracy = " , accuracy_score(y_actual, y_predicted))
    print("F1 Score = " ,f1_score(y_actual, y_predicted))
    pass
generate_model_report(y_test ,y_pred_test)
print(confusion_matrix(y_test ,y_pred_test))
```

```
Accuracy =  0.7058823529411765
F1 Score =  0.6153846153846154
[[80 38]
 [12 40]]
```

In [153]:

```
from sklearn.metrics import roc_curve , auc
y_predict = model.decision_function(X_test)
fpr , tpr , threshold = roc_curve(y_test , y_predict)
auc_L = auc(fpr , tpr)

plt.figure(figsize=(5,5) , dpi = 100)
plt.plot(fpr,tpr ,label = '(auc = %0.3f)'%auc_L)
plt.xlabel('False positive rate')
plt.ylabel('True Positive rate')
plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: