EM_CreditApprovalDataAnalyses

March 11, 2022

1 Erblin Marku

2 Msc Computer Science

3 Queen Mary University of London

4 Data Analytics Coursework 1

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#I will use seaborn library which is build on top of matplotlib for␣
↪vizualisation
import seaborn as sns
# I have a lot of warnings so for visualizations I am going to ignore them.
import warnings
warnings.filterwarnings('ignore')
```

5 1- Data Exploration

[2]:
```
#just loading the main csv without the results table which we will merge later.
crd_df = pd.read_csv('application_record.csv')
```

[3]:
```
#checking the first 10 rows of the dataframe to see how our data is imported
crd_df.head(10)
```

[3]:
```
        ID CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  CNT_CHILDREN  \
0  5008804           M            Y               Y             0
1  5008805           M            Y               Y             0
2  5008806           M            Y               Y             0
3  5008808           F            N               Y             0
4  5008809           F            N               Y             0
5  5008810           F            N               Y             0
6  5008811           F            N               Y             0
7  5008812           F            N               Y             0
8  5008813           F            N               Y             0
```

```
9  5008814           F            N              Y            0

   AMT_INCOME_TOTAL      NAME_INCOME_TYPE            NAME_EDUCATION_TYPE  \
0          427500.0               Working               Higher education
1          427500.0               Working               Higher education
2          112500.0               Working  Secondary / secondary special
3          270000.0  Commercial associate  Secondary / secondary special
4          270000.0  Commercial associate  Secondary / secondary special
5          270000.0  Commercial associate  Secondary / secondary special
6          270000.0  Commercial associate  Secondary / secondary special
7          283500.0             Pensioner               Higher education
8          283500.0             Pensioner               Higher education
9          283500.0             Pensioner               Higher education

     NAME_FAMILY_STATUS  NAME_HOUSING_TYPE  DAYS_BIRTH  DAYS_EMPLOYED  \
0        Civil marriage   Rented apartment      -12005          -4542
1        Civil marriage   Rented apartment      -12005          -4542
2               Married  House / apartment      -21474          -1134
3  Single / not married  House / apartment      -19110          -3051
4  Single / not married  House / apartment      -19110          -3051
5  Single / not married  House / apartment      -19110          -3051
6  Single / not married  House / apartment      -19110          -3051
7             Separated  House / apartment      -22464         365243
8             Separated  House / apartment      -22464         365243
9             Separated  House / apartment      -22464         365243

   FLAG_MOBIL  FLAG_WORK_PHONE  FLAG_PHONE  FLAG_EMAIL OCCUPATION_TYPE  \
0           1                1           0           0             NaN
1           1                1           0           0             NaN
2           1                0           0           0  Security staff
3           1                0           1           1     Sales staff
4           1                0           1           1     Sales staff
5           1                0           1           1     Sales staff
6           1                0           1           1     Sales staff
7           1                0           0           0             NaN
8           1                0           0           0             NaN
9           1                0           0           0             NaN

   CNT_FAM_MEMBERS
0              2.0
1              2.0
2              2.0
3              1.0
4              1.0
5              1.0
6              1.0
7              1.0
```

```
8               1.0
9               1.0
```

[4]:
```
crd_df.info()
#info says we have 18 columns and 438557 entries(rows)
#we see that in the rows we have some data missing in the occupation type since
 ↪we have less entries.
#the others seem ok for now
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 438557 entries, 0 to 438556
Data columns (total 18 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ID                   438557 non-null  int64
 1   CODE_GENDER          438557 non-null  object
 2   FLAG_OWN_CAR         438557 non-null  object
 3   FLAG_OWN_REALTY      438557 non-null  object
 4   CNT_CHILDREN         438557 non-null  int64
 5   AMT_INCOME_TOTAL     438557 non-null  float64
 6   NAME_INCOME_TYPE     438557 non-null  object
 7   NAME_EDUCATION_TYPE  438557 non-null  object
 8   NAME_FAMILY_STATUS   438557 non-null  object
 9   NAME_HOUSING_TYPE    438557 non-null  object
 10  DAYS_BIRTH           438557 non-null  int64
 11  DAYS_EMPLOYED        438557 non-null  int64
 12  FLAG_MOBIL           438557 non-null  int64
 13  FLAG_WORK_PHONE      438557 non-null  int64
 14  FLAG_PHONE           438557 non-null  int64
 15  FLAG_EMAIL           438557 non-null  int64
 16  OCCUPATION_TYPE      304354 non-null  object
 17  CNT_FAM_MEMBERS      438557 non-null  float64
dtypes: float64(2), int64(8), object(8)
memory usage: 60.2+ MB
```

[5]:
```
#first I am going to sort the dataframe by id do get it inline for further
 ↪merging
crd_df = crd_df.sort_values('ID')
crd_df
```

[5]:
```
             ID CODE_GENDER FLAG_OWN_CAR FLAG_OWN_REALTY  CNT_CHILDREN  \
0       5008804           M            Y               Y             0
1       5008805           M            Y               Y             0
2       5008806           M            Y               Y             0
3       5008808           F            N               Y             0
4       5008809           F            N               Y             0
…           …           …            …               …             …
```

```
423317  7999660           F            N              Y           0
426434  7999696           F            N              Y           2
432885  7999738           M            N              Y           0
421225  7999784           F            Y              Y           1
424339  7999952           F            N              Y           1

        AMT_INCOME_TOTAL      NAME_INCOME_TYPE            NAME_EDUCATION_TYPE  \
0               427500.0               Working               Higher education
1               427500.0               Working               Higher education
2               112500.0               Working  Secondary / secondary special
3               270000.0  Commercial associate  Secondary / secondary special
4               270000.0  Commercial associate  Secondary / secondary special
…                    …                    …                             …
423317           90000.0         State servant               Higher education
426434          135000.0         State servant  Secondary / secondary special
432885          180000.0               Working  Secondary / secondary special
421225          180000.0  Commercial associate  Secondary / secondary special
424339          157500.0         State servant               Higher education

          NAME_FAMILY_STATUS  NAME_HOUSING_TYPE  DAYS_BIRTH  DAYS_EMPLOYED  \
0             Civil marriage   Rented apartment      -12005          -4542
1             Civil marriage   Rented apartment      -12005          -4542
2                    Married  House / apartment      -21474          -1134
3       Single / not married  House / apartment      -19110          -3051
4       Single / not married  House / apartment      -19110          -3051
…                        …                  …          …             …
423317  Single / not married  House / apartment      -13432          -5446
426434               Married  House / apartment      -12576          -4382
432885               Married  House / apartment       -9970           -119
421225               Married  House / apartment      -10630           -454
424339               Married  House / apartment      -15859          -3679

        FLAG_MOBIL  FLAG_WORK_PHONE  FLAG_PHONE  FLAG_EMAIL OCCUPATION_TYPE  \
0                1                1           0           0             NaN
1                1                1           0           0             NaN
2                1                0           0           0  Security staff
3                1                0           1           1     Sales staff
4                1                0           1           1     Sales staff
…              …                …          …          …             …
423317           1                0           0           0      Core staff
426434           1                0           0           0  Medicine staff
432885           1                0           0           0             NaN
421225           1                0           0           0             NaN
424339           1                0           0           0      Core staff

        CNT_FAM_MEMBERS
0                   2.0
```

```
1                    2.0
2                    2.0
3                    1.0
4                    1.0
…                  …
423317               1.0
426434               4.0
432885               2.0
421225               3.0
424339               3.0

[438557 rows x 18 columns]
```

5.1 Check if we have duplicated records and remove them to get the uniqe size data

[6]:
```
#check the ID for duplicates and drop those rows by keeping only the last one␣
↪of the duplicated
crd_df.drop_duplicates(subset=['ID'], keep='last', inplace=True)
```

[7]:
```
#lets check the new data size
crd_df.count()
```

[7]:
```
ID                     438510
CODE_GENDER            438510
FLAG_OWN_CAR           438510
FLAG_OWN_REALTY        438510
CNT_CHILDREN           438510
AMT_INCOME_TOTAL       438510
NAME_INCOME_TYPE       438510
NAME_EDUCATION_TYPE    438510
NAME_FAMILY_STATUS     438510
NAME_HOUSING_TYPE      438510
DAYS_BIRTH             438510
DAYS_EMPLOYED          438510
FLAG_MOBIL             438510
FLAG_WORK_PHONE        438510
FLAG_PHONE             438510
FLAG_EMAIL             438510
OCCUPATION_TYPE        304322
CNT_FAM_MEMBERS        438510
dtype: int64
```

[8]:
```
#generate discriptive statistics from our data with describe() function
crd_df.describe()
#added this code to change the number from scientific notation to numbers
pd.set_option('display.float_format', lambda x: '%.2f' % x)
```

```
#we can also transpose the table for better view
crd_df.describe()
#crd_df.describe().T
```

[8]:

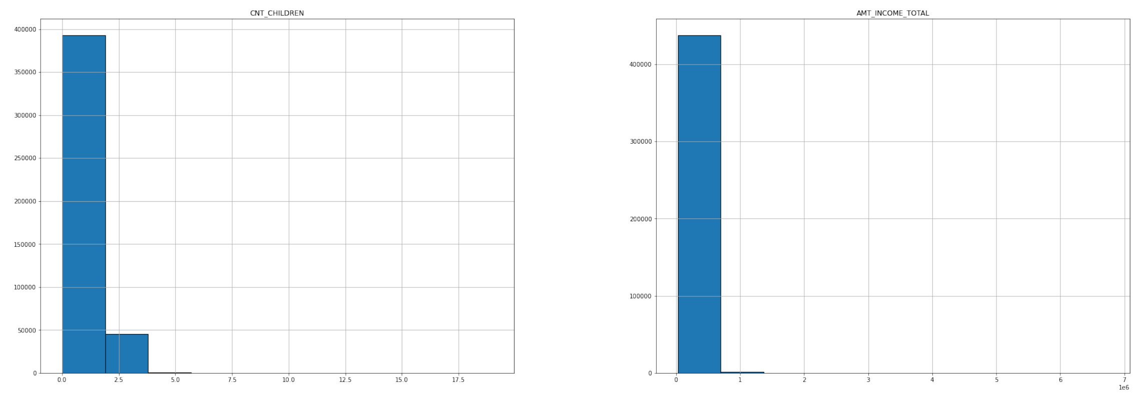
```
                ID  CNT_CHILDREN  AMT_INCOME_TOTAL  DAYS_BIRTH  DAYS_EMPLOYED  \
count  438510.00     438510.00         438510.00   438510.00      438510.00
mean  6022034.96          0.43         187524.26   -15997.89       60561.99
std    571496.24          0.72         110087.41     4185.00      138766.43
min   5008804.00          0.00          26100.00   -25201.00      -17531.00
25%   5609362.25          0.00         121500.00   -19483.00       -3103.00
50%   6047719.50          0.00         161100.00   -15630.00       -1468.00
75%   6454160.75          1.00         225000.00   -12514.00        -371.00
max   7999952.00         19.00        6750000.00    -7489.00      365243.00

       FLAG_MOBIL  FLAG_WORK_PHONE  FLAG_PHONE  FLAG_EMAIL  CNT_FAM_MEMBERS
count   438510.00        438510.00   438510.00   438510.00        438510.00
mean         1.00             0.21        0.29        0.11             2.19
std          0.00             0.40        0.45        0.31             0.90
min          1.00             0.00        0.00        0.00             1.00
25%          1.00             0.00        0.00        0.00             2.00
50%          1.00             0.00        0.00        0.00             2.00
75%          1.00             0.00        1.00        0.00             3.00
max          1.00             1.00        1.00        1.00            20.00
```

6 2- Visualization

[9]:

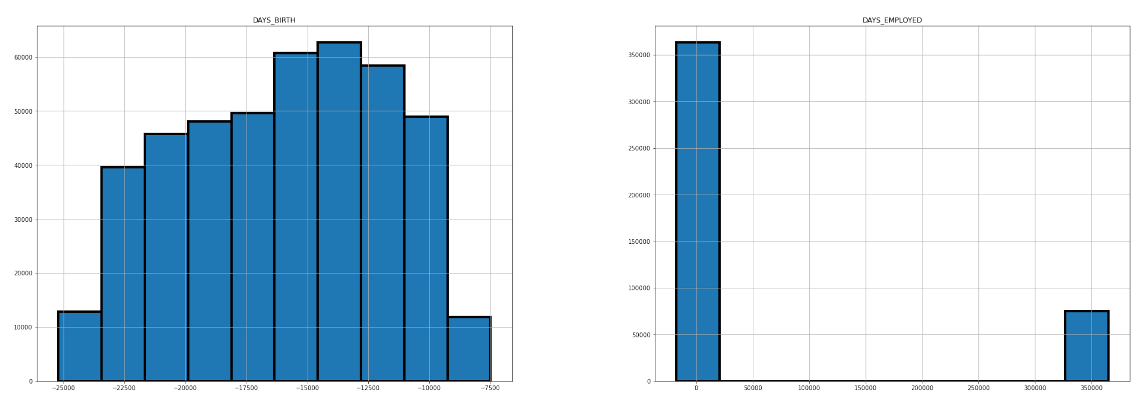
```
plt.figure(figsize=(8,8))
#I will check children count, total income, age and employment days to see any␣
↪negative patterns
# these plots are the ones I can check numerically, the others I will have to␣
↪plot different type of tables
cols_to_plot = ["CNT_CHILDREN","AMT_INCOME_TOTAL"]
crd_df[cols_to_plot].hist(edgecolor='black', linewidth=1)
dev_check=plt.gcf()#get current figure, if no figure create a new one
dev_check.set_size_inches(33,11)
```

```
<Figure size 576x576 with 0 Axes>
```

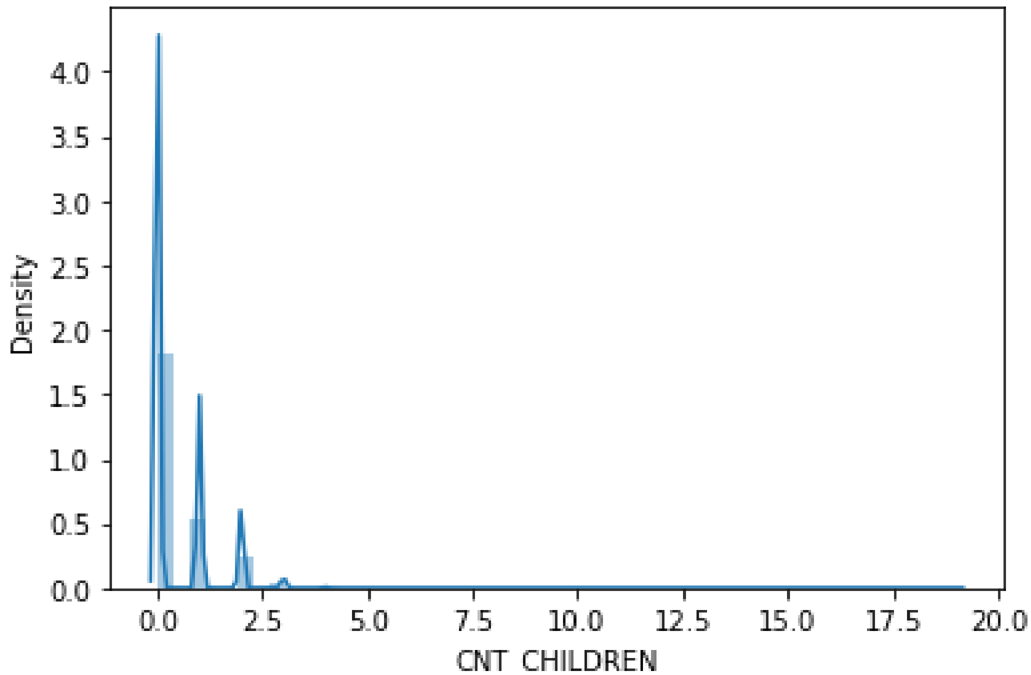


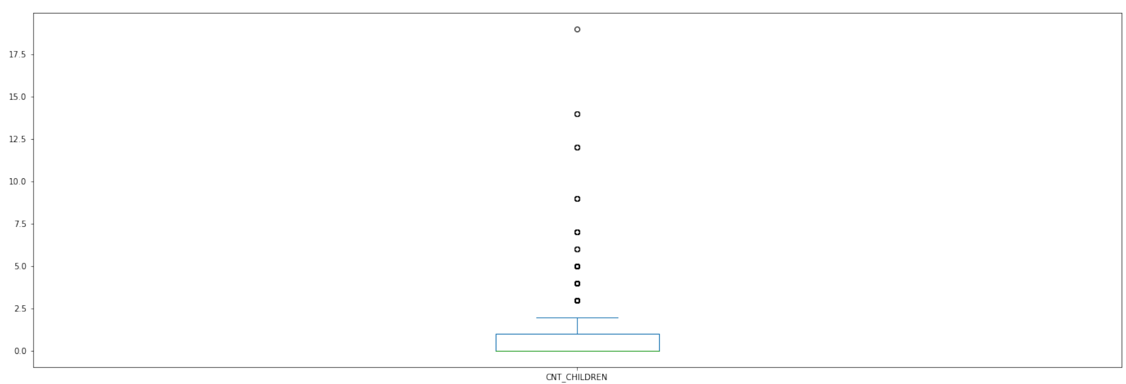
[10]:
```
plt.figure(figsize=(8,8))
#I will check children count, total income, age and employment days to see any negative patterns
# these plots are the ones I can check numerically, the others I will have to plot different type of tables
cols_to_plot = ["DAYS_BIRTH","DAYS_EMPLOYED"]
crd_df[cols_to_plot].hist(edgecolor='black', linewidth=4)
dev_check=plt.gcf()
dev_check.set_size_inches(33,11)
```

```
<Figure size 576x576 with 0 Axes>
```



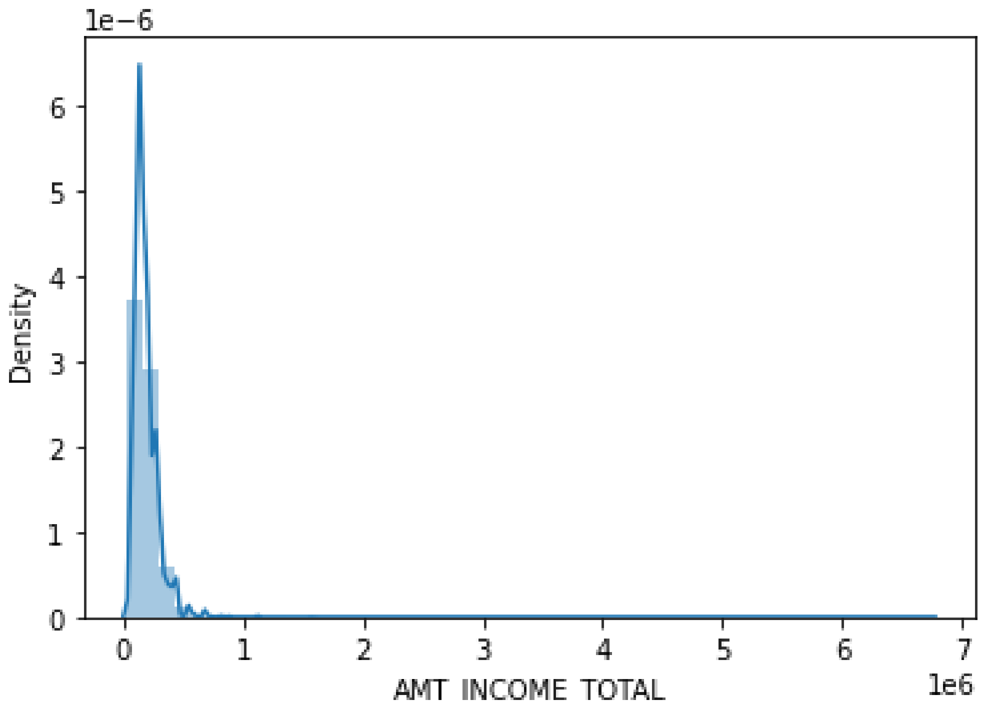
[11]:
```
sns.distplot(crd_df["CNT_CHILDREN"])#seaborn distribution plot
plt.show()
crd_df["CNT_CHILDREN"].plot.box(figsize=(24,8))#box plot to show outliners. Outliners are the dots more far away from the boxes
plt.show()
```

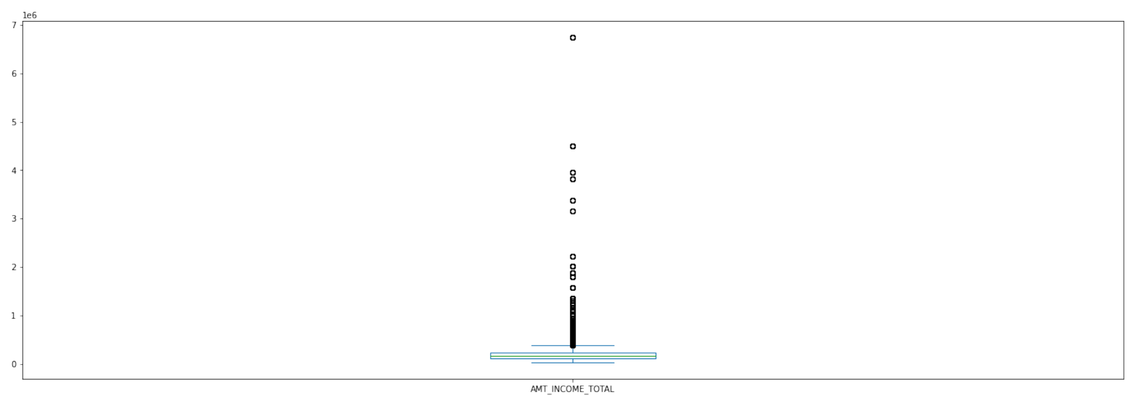




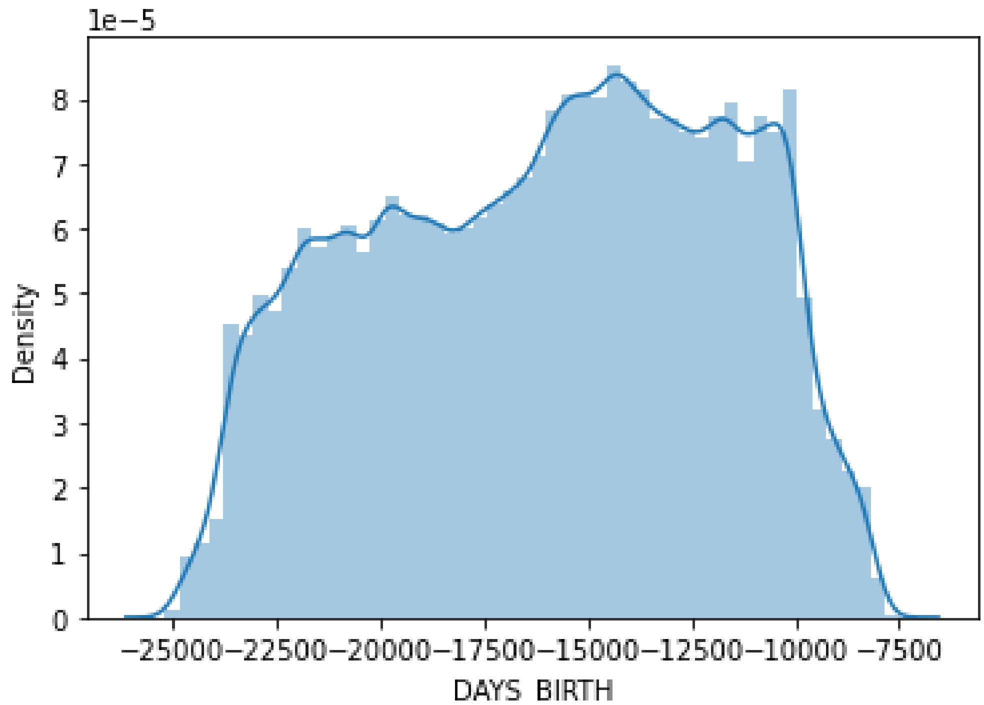
[12]:
```
#by the box plot we see some outliners in this category which I will remove later
```

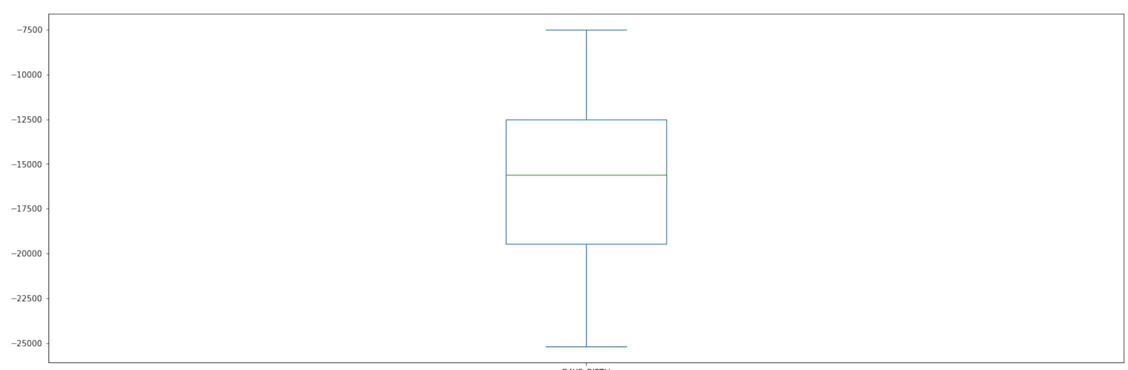
[13]:
```
sns.distplot(crd_df["AMT_INCOME_TOTAL"])
plt.show()
crd_df["AMT_INCOME_TOTAL"].plot.box(figsize=(24,8))
plt.show()
```



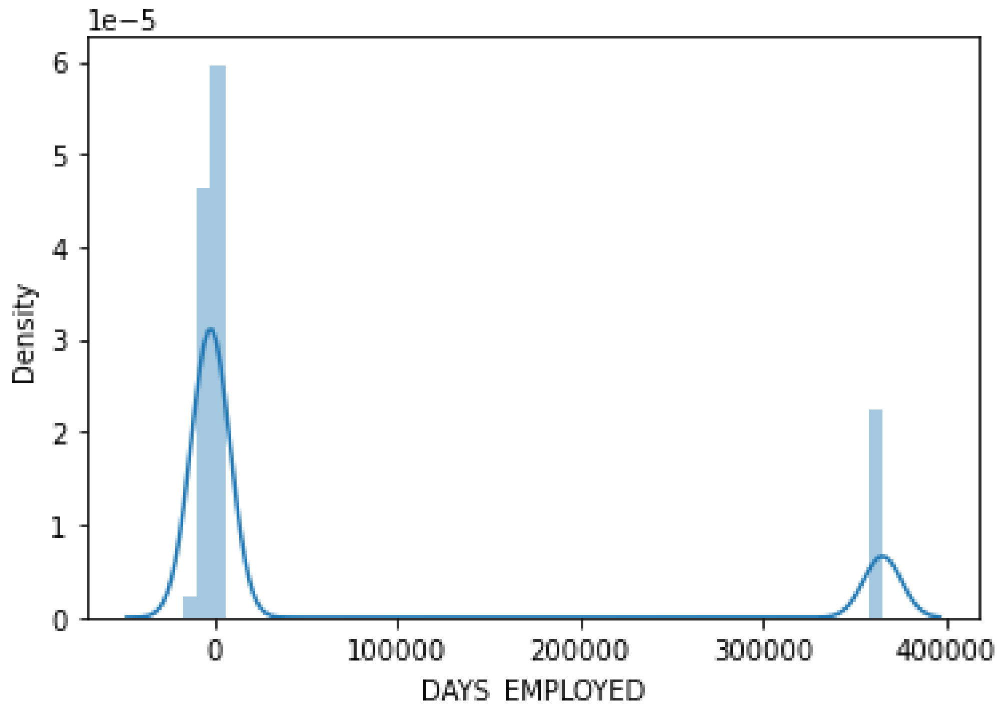


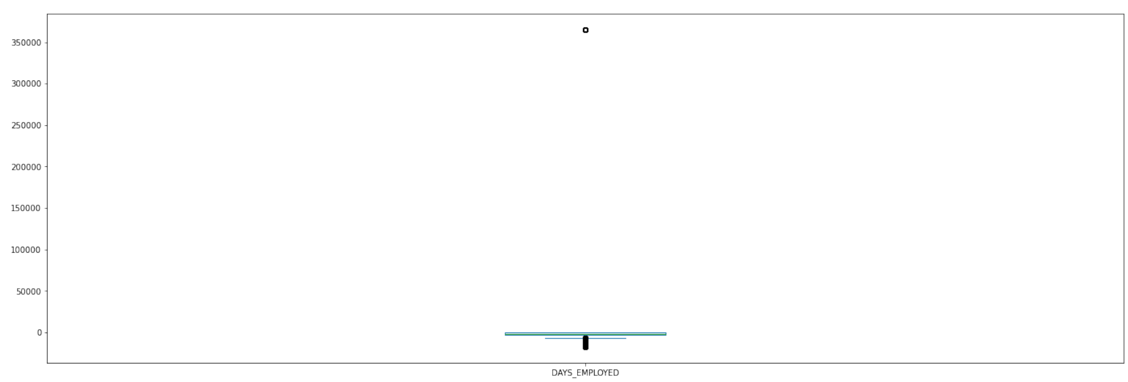
[14]:
```
sns.distplot(crd_df["DAYS_BIRTH"])
plt.show()
crd_df["DAYS_BIRTH"].plot.box(figsize=(24,8))
plt.show()
```





[15]:
```
sns.distplot(crd_df["DAYS_EMPLOYED"])
plt.show()
crd_df["DAYS_EMPLOYED"].plot.box(figsize=(24,8))
plt.show()
```


1e−5
6
5
4
3
2
1
0
Density
0
100000
200000
300000
400000
DAYS_EMPLOYED


350000
300000
250000
200000
150000
100000
50000
0
DAYS_EMPLOYED

6.0.1 Now I see some problems in the number of children data, days employed and total income graphs. We can spot outliners from here.

6.0.2 This means I will have to drop some rows with high number of children and the hole employed graph

6.0.3 I need also to convert days employed and age into years.

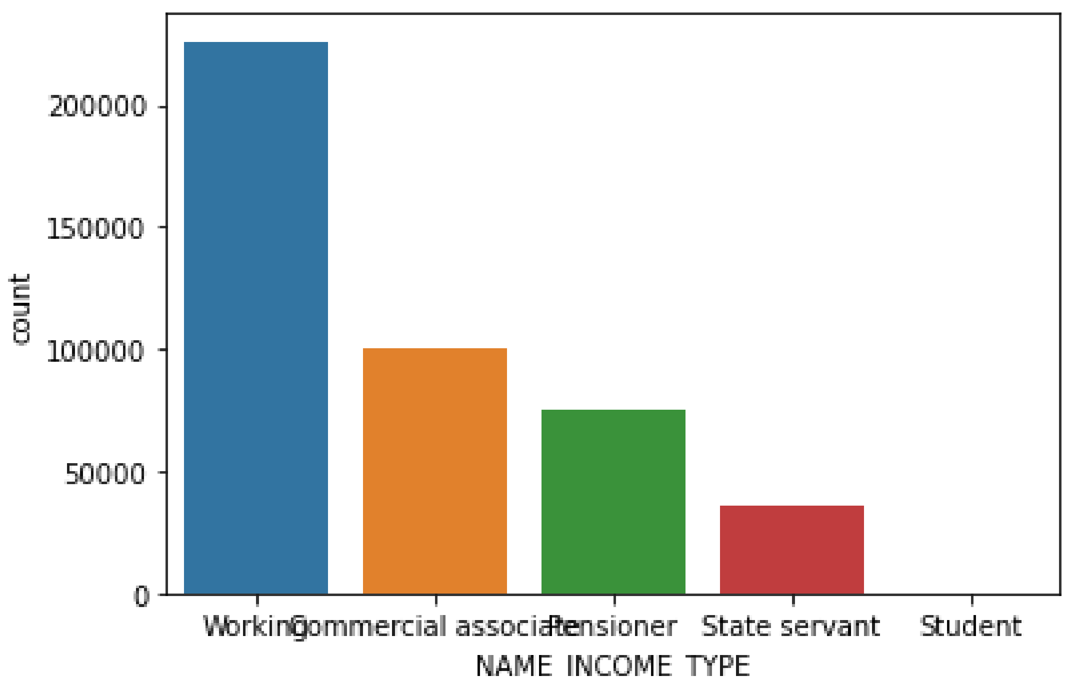
7 Lets plot the non-numeric value columns

I will plot the income type,family status,housing type and education type with hist charts

The Gender, car and realty ownerships will be ploted with pie charts because they are Yes/No values, I will try also 1/0 values

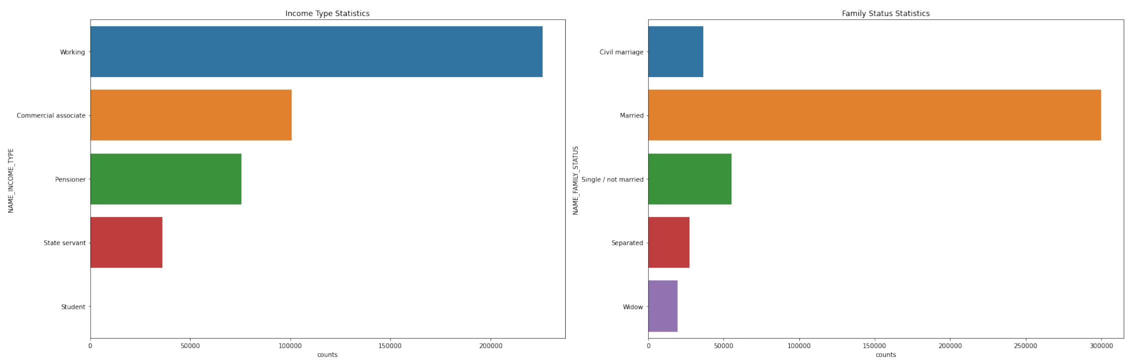
[16]:
```
sns.countplot(crd_df['NAME_INCOME_TYPE'])# Similarly, we can visualise the␣
↪distribution of the numerical variables
```

[16]:
```
<AxesSubplot:xlabel='NAME_INCOME_TYPE', ylabel='count'>
```



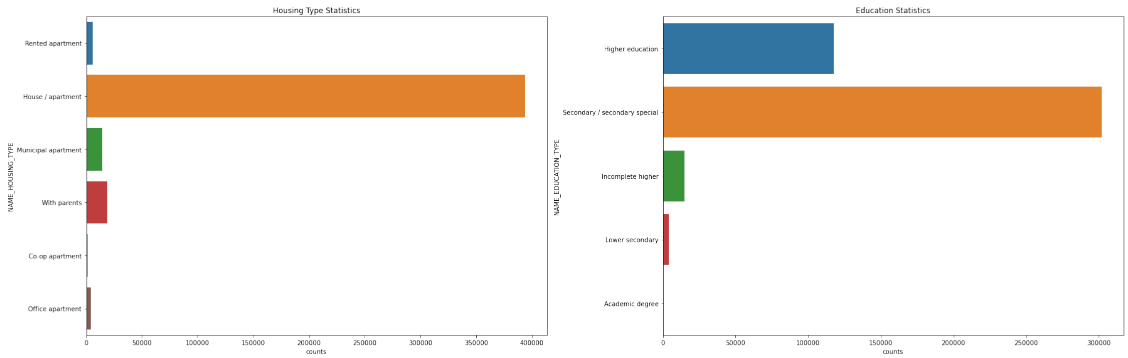
[17]:
```
#Income type and family status charts
#I am using counts of theem to see how may entries are in each category
fig, axes = plt.subplots(1,2)

f1=sns.countplot(y=crd_df.NAME_INCOME_TYPE,linewidth=1.5, ax=axes[0])
f1.set_title("Income Type Statistics")
```

```
f1.set_xlabel("counts")

f2=sns.countplot(y=crd_df.NAME_FAMILY_STATUS,linewidth=1.5, ax=axes[1])
f2.set_title("Family Status Statistics")
f2.set_xlabel("counts")


fig.set_size_inches(25,8)
plt.tight_layout()
plt.show()
```



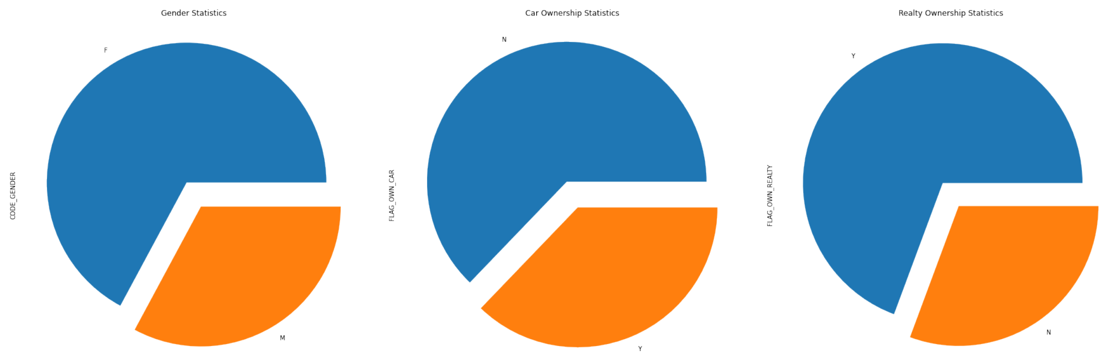
7.0.1 Student seems to have less importance, but the occupation is a must when have credit approval so we will have to keep it for now

[18]:
```
fig, axes = plt.subplots(1,2)

f1= sns.countplot(y=crd_df.NAME_HOUSING_TYPE,linewidth=1.5, ax=axes[0])
f1.set_title("Housing Type Statistics")
f1.set_xlabel("counts")


f2= sns.countplot(y=crd_df.NAME_EDUCATION_TYPE, ax=axes[1])
f2.set_title("Education Statistics")
f2.set_xlabel("counts")

fig.set_size_inches(25,8)

plt.tight_layout()

plt.show()
```



7.0.2 Housing type also has a large value of house and apartments, which may effect our learning but we will have to keep it since it is a relevant information for credit check

[19]:
```
#plot the pie charts for the yes/no value columns
fig, axes = plt.subplots(1,3)

f1= crd_df['CODE_GENDER'].value_counts().plot.pie(explode=[0.1,0.1], ax=axes[0])
f1.set_title("Gender Statistics")

f2= crd_df['FLAG_OWN_CAR'].value_counts().plot.pie(explode=[0.1,0.1], ax=axes[1])
f2.set_title("Car Ownership Statistics")

f3= crd_df['FLAG_OWN_REALTY'].value_counts().plot.pie(explode=[0.1,0.1], ax=axes[2])
f3.set_title("Realty Ownership Statistics")


fig.set_size_inches(25,8)

plt.tight_layout()

plt.show()
```

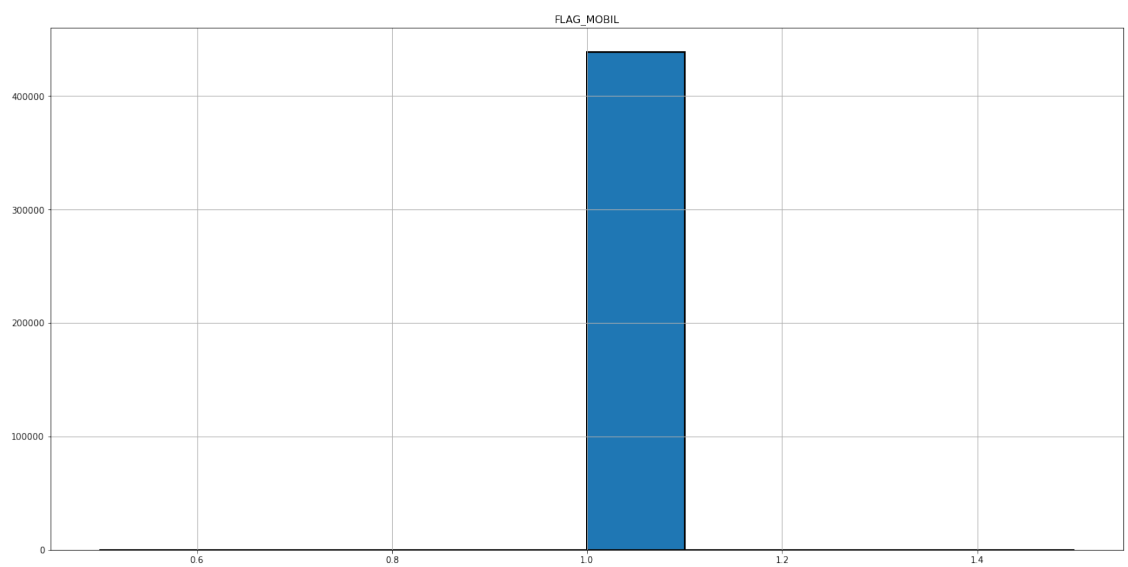


[20]:
```
crd_df['FLAG_MOBIL'].unique()
#checking the values of mobile phone
#we can see we have only 1 value so it is redundant for our model and we will drop it later.
```

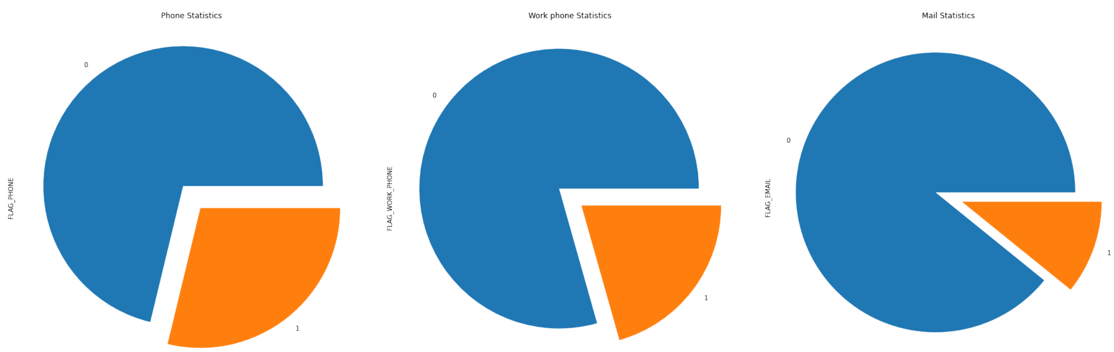
[20]:
```
array([1])
```

[21]:
```
#plot the hist charts to see the 1/0 values also
plt.figure(figsize=(8,8))

cols_to_plot = ["FLAG_MOBIL"]
crd_df[cols_to_plot].hist(edgecolor='black', linewidth=2)
dev_check=plt.gcf()
dev_check.set_size_inches(22,11)
```

```
<Figure size 576x576 with 0 Axes>
```



[22]:
```
#plot the pie charts for the yes/no value columns
fig, axes = plt.subplots(1,3)

#f1= crd_df['FLAG_MOBIL'].value_counts().plot.pie(explode=[0.1,0.1], ax=axes[0])
#f1.set_title("Mobile Statistics")

f1= crd_df['FLAG_PHONE'].value_counts().plot.pie(explode=[0.1,0.1], ax=axes[0])
f1.set_title("Phone Statistics")

f2= crd_df['FLAG_WORK_PHONE'].value_counts().plot.pie(explode=[0.1,0.1],
    ax=axes[1])
f2.set_title("Work phone Statistics")

f3= crd_df['FLAG_EMAIL'].value_counts().plot.pie(explode=[0.1,0.1], ax=axes[2])
f3.set_title("Mail Statistics")


fig.set_size_inches(25,8)

plt.tight_layout()

plt.show()
```



8 3- Pre-processing

8.1 *Converting categorical values to ones and zeros

[23]:
```
crd_df['CODE_GENDER'].unique()
#checking the values of gender
```

[23]:
```
array(['M', 'F'], dtype=object)
```

[24]:
```
#replace the values by using a dicionary with new values for previous keys.
#male will be 1 and female 0, this does not matter as we only need to differ
crd_df = crd_df.replace(
                            {'CODE_GENDER' :
                                 {'M':1, 'F':0}
                            }
                        )
crd_df['FLAG_OWN_REALTY'].unique()
```

[24]:
```
array(['Y', 'N'], dtype=object)
```

[25]:
```
crd_df['FLAG_OWN_CAR'].unique()
```

[25]:
```
array(['Y', 'N'], dtype=object)
```

[26]:
```
crd_df = crd_df.replace(
                            {'FLAG_OWN_CAR' :
                                        {'Y':1,'N':0}
                            }
                        )
crd_df['FLAG_OWN_CAR'].unique()
```

[26]:
```
array([1, 0])
```

[27]:
```
crd_df['FLAG_OWN_REALTY'].unique()
```

[27]:
```
array(['Y', 'N'], dtype=object)
```

[28]:
```
crd_df = crd_df.replace({'FLAG_OWN_REALTY' :
                                            {'Y' : 1,
                                             'N' : 0}})
crd_df['FLAG_OWN_REALTY'].unique()
```

[28]:
```
array([1, 0])
```

[29]:
```
#Education type and Family status have a lot of categories, so I will use an␣
 ↪encoder for those.
crd_df['NAME_EDUCATION_TYPE'].unique()
```

[29]:
```
array(['Higher education', 'Secondary / secondary special',
       'Incomplete higher', 'Lower secondary', 'Academic degree'],
      dtype=object)
```

[30]:
```
crd_df['NAME_FAMILY_STATUS'].unique()
```

[30]:
```
array(['Civil marriage', 'Married', 'Single / not married', 'Separated',
       'Widow'], dtype=object)
```

[31]:
```
crd_df['CNT_FAM_MEMBERS'].unique()
```

[31]:
```
array([ 2.,  1.,  5.,  3.,  4.,  6., 15.,  7., 20.,  9., 11., 14.,  8.])
```

[32]:
```
crd_df['CNT_FAM_MEMBERS'] = crd_df['CNT_FAM_MEMBERS'].astype(int)
crd_df['CNT_FAM_MEMBERS'].unique()
```

[32]:
```
array([ 2,  1,  5,  3,  4,  6, 15,  7, 20,  9, 11, 14,  8])
```

8.2 *Converting the number of days format in DAYS_BIRTH and DAYS_EMPLOYED to number of years

[33]:
```
# Using pandas timedelta type, this helps me in the conversion of days to years
crd_df['AGE'] = np.ceil(pd.to_timedelta(crd_df['DAYS_BIRTH'], unit='D').dt.days␣
 ↪/ -365.25)
#lets check the conversion
crd_df['AGE']
```

[33]:
```
0         33.00
1         33.00
2         59.00
3         53.00
4         53.00
           …
423317   37.00
426434   35.00
432885   28.00
421225   30.00
424339   44.00
Name: AGE, Length: 438510, dtype: float64
```

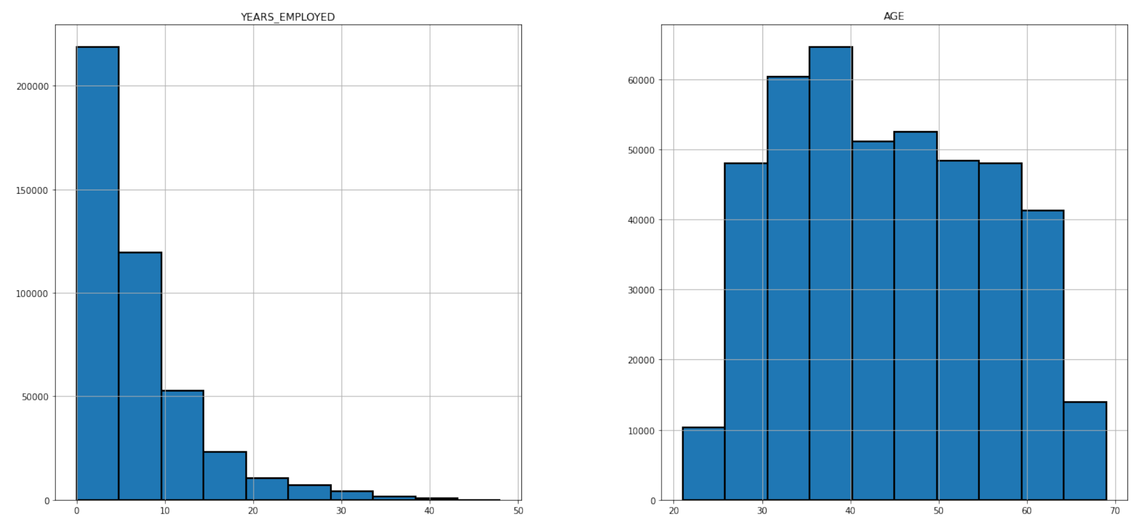
[34]:
```
#I need to get rid of values more than 0 wich means they are not working.
#so I will convert them all to 0
crd_df.loc[(crd_df['DAYS_EMPLOYED'] > 0), 'DAYS_EMPLOYED'] = 0
```

[35]:
```
#convert the same way we did with age
crd_df['YEARS_EMPLOYED'] = np.ceil(pd.to_timedelta(crd_df['DAYS_EMPLOYED'],␣
 ↪unit='D').dt.days / -365.25)
crd_df['YEARS_EMPLOYED'].unique()
```

[35]:
```
array([13.,  4.,  9., -0.,  3.,  5.,  6., 20., 15., 14.,  8.,  7., 18.,
       30.,  2., 16., 12.,  1., 11., 24., 25., 21., 10., 28., 27., 19.,
       22., 23., 17., 29., 39., 33., 32., 37., 38., 31., 40., 26., 35.,
       34., 42., 41., 36., 44., 43., 45., 48., 46.])
```

[36]:
```python
#Lets see their grapghs now
plt.figure(figsize=(8,8))

cols_to_plot = ["YEARS_EMPLOYED","AGE"]
crd_df[cols_to_plot].hist(edgecolor='black', linewidth=2)
dev_check=plt.gcf()
dev_check.set_size_inches(22,10)
```

```
<Figure size 576x576 with 0 Axes>
```



8.3 *Calculate the z-scores to see and remove outliners, so values that are far from my mean.

I will do this in 3 columns which seem suspicious by their charts and data and logically. These being number of children, total income and years employed.

[37]:
```python
def calc_z_score(df, cols):
    for col in cols:
        df[col+"_z_score"] = (df[col] - df[col].mean())/df[col].std()
    return df

crd_df_2 = calc_z_score(crd_df, cols=["CNT_CHILDREN","AMT_INCOME_TOTAL",␣
↪"YEARS_EMPLOYED"])

print(crd_df_2['CNT_CHILDREN_z_score'])
print(crd_df_2['AMT_INCOME_TOTAL_z_score'])
print(crd_df_2['YEARS_EMPLOYED_z_score'])
```

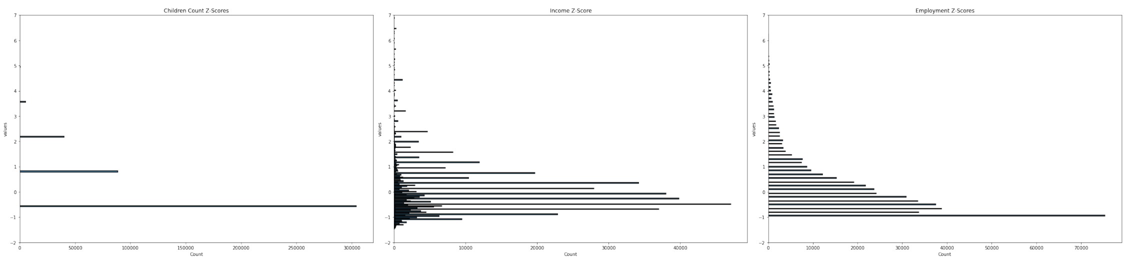
```
0         -0.59
1         -0.59
2         -0.59
```

```
3         -0.59
4         -0.59
           …
423317   -0.59
426434    2.17
432885   -0.59
421225    0.79
424339    0.79
Name: CNT_CHILDREN_z_score, Length: 438510, dtype: float64
0         2.18
1         2.18
2        -0.68
3         0.75
4         0.75
           …
423317   -0.89
426434   -0.48
432885   -0.07
421225   -0.07
424339   -0.27
Name: AMT_INCOME_TOTAL_z_score, Length: 438510, dtype: float64
0         1.00
1         1.00
2        -0.35
3         0.40
4         0.40
           …
423317    1.30
426434    0.85
432885   -0.81
421225   -0.66
424339    0.70
Name: YEARS_EMPLOYED_z_score, Length: 438510, dtype: float64
```

[38]:

```
#lets see also their histplots to make a correct assumption of the threshold.
fig, axes = plt.subplots(1,3)

f1=sns.histplot(y=crd_df_2.CNT_CHILDREN_z_score,linewidth=1.5, ax=axes[0])
f1.set_title("Children Count Z-Scores")
f1.set_ylabel("values")
f1.set_xlabel("Count")
f1.set_ylim([-2,7])

f2= sns.histplot(y=crd_df_2.AMT_INCOME_TOTAL_z_score,linewidth=1.5, ax=axes[1])
f2.set_title("Income Z-Score")
f2.set_ylabel("values")
f2.set_ylim([-2,7])
```

```
f3= sns.histplot(y=crd_df_2.YEARS_EMPLOYED_z_score,linewidth=1.5, ax=axes[2])
f3.set_title("Employment Z-Scores")
f3.set_ylabel("values")
f3.set_ylim([-2,7])

fig.set_size_inches(35,8)
f1.set_ylim([-2,7])
plt.tight_layout()

plt.show()
```



[39]:
```
#By the data and charts it seem good to use the empirical rule, so I am going␣
↪to keep the threshold at 3 to get
#I use 3 filters and apply it to the new df while checking the absolute value␣
↪of the z-score and making sure the z-score is between -3 and 3.
filter_2 = crd_df_2.CNT_CHILDREN_z_score.abs() <= 3
filter_3 = crd_df_2.AMT_INCOME_TOTAL_z_score.abs() <= 3
filter_4 = crd_df_2.YEARS_EMPLOYED_z_score.abs() <= 3

crd_df_2 = crd_df_2[filter_2 & filter_3 & filter_4]

crd_df_2.drop(columns=␣
↪["CNT_CHILDREN_z_score","AMT_INCOME_TOTAL_z_score","YEARS_EMPLOYED_z_score"],inplace=True)
```

8.4 *Drop null value rows found in OCCUPATION TYPE from the dataframe

[40]:
```
crd_clean = crd_df_2[crd_df_2['OCCUPATION_TYPE'].notna()]
```

[41]:
```
crd_clean.count()
#we can see that the rows are unified and we have succesfully dropped the rows
```

[41]:
```
ID                     287950
CODE_GENDER            287950
FLAG_OWN_CAR           287950
FLAG_OWN_REALTY        287950
```

```
CNT_CHILDREN          287950
AMT_INCOME_TOTAL      287950
NAME_INCOME_TYPE      287950
NAME_EDUCATION_TYPE   287950
NAME_FAMILY_STATUS    287950
NAME_HOUSING_TYPE     287950
DAYS_BIRTH            287950
DAYS_EMPLOYED         287950
FLAG_MOBIL            287950
FLAG_WORK_PHONE       287950
FLAG_PHONE            287950
FLAG_EMAIL            287950
OCCUPATION_TYPE       287950
CNT_FAM_MEMBERS       287950
AGE                   287950
YEARS_EMPLOYED        287950
dtype: int64
```

8.5 *Remove Mobile label

[42]:
```
crd_df = crd_df.drop(columns=['FLAG_MOBIL'])
```

8.6 *Renaming columns and dropping redundant features that we already converted

[43]:
```
crd_clean = crd_clean.drop(columns=['DAYS_BIRTH', 'DAYS_EMPLOYED'])
```

[44]:
```
#Rename the columns for better view
crd_clean.columns = ['ID', 'Gender', 'Car_Owner', 'Realty_Owner', 'Children',␣
↪'Total_Income', 'Income_Type',\
                        'Education_Type', 'Family_Status', 'Housing_Type',␣
↪'Mobile_Phone', 'Work_Phone', \
                        'Phone', 'Email', 'Job_Title', 'Family_Members', 'Age',␣
↪'Years_Experience' ]
```

[45]:
```
crd_clean.head()
```

[45]:
```
        ID  Gender  Car_Owner  Realty_Owner  Children  Total_Income  \
2  5008806       1          1             1         0     112500.00
3  5008808       0          0             1         0     270000.00
4  5008809       0          0             1         0     270000.00
5  5008810       0          0             1         0     270000.00
6  5008811       0          0             1         0     270000.00

           Income_Type                 Education_Type         Family_Status  \
2              Working  Secondary / secondary special               Married
```

```
3  Commercial associate  Secondary / secondary special  Single / not married
4  Commercial associate  Secondary / secondary special  Single / not married
5  Commercial associate  Secondary / secondary special  Single / not married
6  Commercial associate  Secondary / secondary special  Single / not married

        Housing_Type  Mobile_Phone  Work_Phone  Phone  Email       Job_Title  \
2  House / apartment             1           0      0      0  Security staff
3  House / apartment             1           0      1      1     Sales staff
4  House / apartment             1           0      1      1     Sales staff
5  House / apartment             1           0      1      1     Sales staff
6  House / apartment             1           0      1      1     Sales staff

   Family_Members   Age  Years_Experience
2               2 59.00              4.00
3               1 53.00              9.00
4               1 53.00              9.00
5               1 53.00              9.00
6               1 53.00              9.00
```

9 *Loading the second csv and make some preprocessing

[46]:
```
crdr_df = pd.read_csv('credit_record.csv')
```

[47]:
```
crdr_df.head()
```

[47]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

[48]:
```
#lets sort it by ID to get inline with the first dataframe.
crdr_df = crdr_df.sort_values('ID')
crdr_df
```

[48]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
22	5001712	-18	0
…	…	…	…
1048547	5150487	-2	C
1048546	5150487	-1	C
1048545	5150487	0	C

```
1048558  5150487             -13      C
1048574  5150487             -29      C

[1048575 rows x 3 columns]
```

[49]:
```
#I will check the different types of status to classify the data manually
crdr_df['STATUS'].unique()
```

[49]:
```
array(['X', '0', 'C', '1', '3', '2', '4', '5'], dtype=object)
```

[50]:
```
crdr_df['STATUS'].value_counts()
```

[50]:
```
C    442031
0    383120
X    209230
1     11090
5      1693
2       868
3       320
4       223
Name: STATUS, dtype: int64
```

10 From the information in the dataset I will classify debt as below

C, X, 0 with 'Good_Debt' (C: loan for that month is already paid; X: no loan for that month; 0: loan is 1 to 29 days overdue).

1, 2, 3, 4, 5 with 'Bad_Debt' (1: loan is 30 to 59 days overdue; 2: loan is 60 to 89 days overdue; 3: loan is 90 to 119 days overdue; 4: loan is 120 to 149 days overdue; 5: loan is more than 150 days overdue).

[51]:
```
crdr_df['STATUS_NEW'] = crdr_df['STATUS']
```

[52]:
```
crdr_df = crdr_df.replace({'STATUS_NEW' :
                                            {'C' : 'Good_Debt',
                                             'X' : 'Good_Debt',
                                             '0' : 'Good_Debt',
                                             '1' : 'Bad_Debt',
                                             '2' : 'Bad_Debt',
                                             '3' : 'Bad_Debt',
                                             '4' : 'Bad_Debt',
                                             '5' : 'Bad_Debt'}})
```

[53]:
```
crdr_df
```

[53]:
```
              ID  MONTHS_BALANCE STATUS STATUS_NEW
0        5001711               0      X  Good_Debt
```

```
1         5001711             -1     0  Good_Debt
2         5001711             -2     0  Good_Debt
3         5001711             -3     0  Good_Debt
22        5001712            -18     0  Good_Debt
…           …           …     …       …
1048547   5150487             -2     C  Good_Debt
1048546   5150487             -1     C  Good_Debt
1048545   5150487              0     C  Good_Debt
1048558   5150487            -13     C  Good_Debt
1048574   5150487            -29     C  Good_Debt

[1048575 rows x 4 columns]
```

[54]:

```
crdr_df['STATUS'].value_counts()
```

[54]:

```
C    442031
0    383120
X    209230
1     11090
5      1693
2       868
3       320
4       223
Name: STATUS, dtype: int64
```

[55]:

```
# I can see multiple debts for one ID, so I need to compile total number of Bad␣
↪and Good debts for each client(ID)
crdr_df.value_counts(subset=['ID', 'STATUS_NEW']).unstack(fill_value=0)
```

[55]:

```
STATUS_NEW  Bad_Debt  Good_Debt
ID
5001711            0          4
5001712            0         19
5001713            0         22
5001714            0         15
5001715            0         60
…                …         …
5150482            0         18
5150483            0         18
5150484            0         13
5150485            0          2
5150487            0         30

[45985 rows x 2 columns]
```

10.1 Classify clients according to the number of bad an good debts.

[56]:
```
#first I will create and index fo the dataset
res_df = crdr_df.value_counts(subset=['ID', 'STATUS_NEW']).
↪unstack(fill_value=0).reset_index()
```

[57]:
```
res_df
```

[57]:
```
STATUS_NEW       ID  Bad_Debt  Good_Debt
0           5001711         0          4
1           5001712         0         19
2           5001713         0         22
3           5001714         0         15
4           5001715         0         60
…               …         …         …
45980       5150482         0         18
45981       5150483         0         18
45982       5150484         0         13
45983       5150485         0          2
45984       5150487         0         30

[45985 rows x 3 columns]
```

[58]:
```
#For the comparison I am building an user facilitator function, which means I␣
↪will approve those with same good and bad depts and reject those with more␣
↪bad debts than good.
res_df.loc[(res_df['Good_Debt'] >= res_df['Bad_Debt']), 'Status'] = 1
res_df.loc[(res_df['Good_Debt'] < res_df['Bad_Debt']), 'Status'] = 0
res_df['Status'] = res_df['Status'].astype(int)
```

[59]:
```
res_df.tail()
```

[59]:
```
STATUS_NEW       ID  Bad_Debt  Good_Debt  Status
45980       5150482         0         18       1
45981       5150483         0         18       1
45982       5150484         0         13       1
45983       5150485         0          2       1
45984       5150487         0         30       1
```

[60]:
```
res_df['Status'].value_counts()
```

[60]:
```
1    45847
0      138
Name: Status, dtype: int64
```

11 Merge the Results with the first df by ID

[61]:
```
df = crd_clean.merge(res_df, how='inner', on=['ID'])
```

[62]:
```
df.head()
```

[62]:
```
        ID  Gender  Car_Owner  Realty_Owner  Children  Total_Income  \
0  5008806       1          1             1         0     112500.00
1  5008808       0          0             1         0     270000.00
2  5008809       0          0             1         0     270000.00
3  5008810       0          0             1         0     270000.00
4  5008811       0          0             1         0     270000.00

            Income_Type                  Education_Type          Family_Status  \
0               Working  Secondary / secondary special                Married
1  Commercial associate  Secondary / secondary special  Single / not married
2  Commercial associate  Secondary / secondary special  Single / not married
3  Commercial associate  Secondary / secondary special  Single / not married
4  Commercial associate  Secondary / secondary special  Single / not married

        Housing_Type  …  Work_Phone  Phone  Email       Job_Title  \
0  House / apartment  …           0      0      0  Security staff
1  House / apartment  …           0      1      1     Sales staff
2  House / apartment  …           0      1      1     Sales staff
3  House / apartment  …           0      1      1     Sales staff
4  House / apartment  …           0      1      1     Sales staff

   Family_Members   Age  Years_Experience  Bad_Debt  Good_Debt  Status
0               2 59.00              4.00         0         30       1
1               1 53.00              9.00         0          5       1
2               1 53.00              9.00         0          5       1
3               1 53.00              9.00         0         27       1
4               1 53.00              9.00         0         39       1

[5 rows x 21 columns]
```

[63]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23906 entries, 0 to 23905
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID                 23906 non-null  int64
 1   Gender             23906 non-null  int64
 2   Car_Owner          23906 non-null  int64
 3   Realty_Owner       23906 non-null  int64
```

```
 4   Children          23906 non-null  int64
 5   Total_Income      23906 non-null  float64
 6   Income_Type       23906 non-null  object
 7   Education_Type    23906 non-null  object
 8   Family_Status     23906 non-null  object
 9   Housing_Type      23906 non-null  object
 10  Mobile_Phone      23906 non-null  int64
 11  Work_Phone        23906 non-null  int64
 12  Phone             23906 non-null  int64
 13  Email             23906 non-null  int64
 14  Job_Title         23906 non-null  object
 15  Family_Members    23906 non-null  int64
 16  Age               23906 non-null  float64
 17  Years_Experience  23906 non-null  float64
 18  Bad_Debt          23906 non-null  int64
 19  Good_Debt         23906 non-null  int64
 20  Status            23906 non-null  int64
dtypes: float64(3), int64(13), object(5)
memory usage: 4.0+ MB
```

[64]:
```
df = df.drop(columns=['Mobile_Phone'])
```

[65]:
```
df['Status'].value_counts()
```

[65]:
```
1    23824
0       82
Name: Status, dtype: int64
```

[66]:
```
# create lists of numerical and categorical variables to use for graphs
catg_vars = [var for var in df.columns if var != 'Status' and df[var].
 ↪dtype=='O']
num_vars = [var for var in df.columns if var != 'Status' and var not in␣
 ↪catg_vars]

print('Number of categorical variables: {}'.format(len(catg_vars)))
print('Number of numerical variables: {}'.format(len(num_vars)))
```

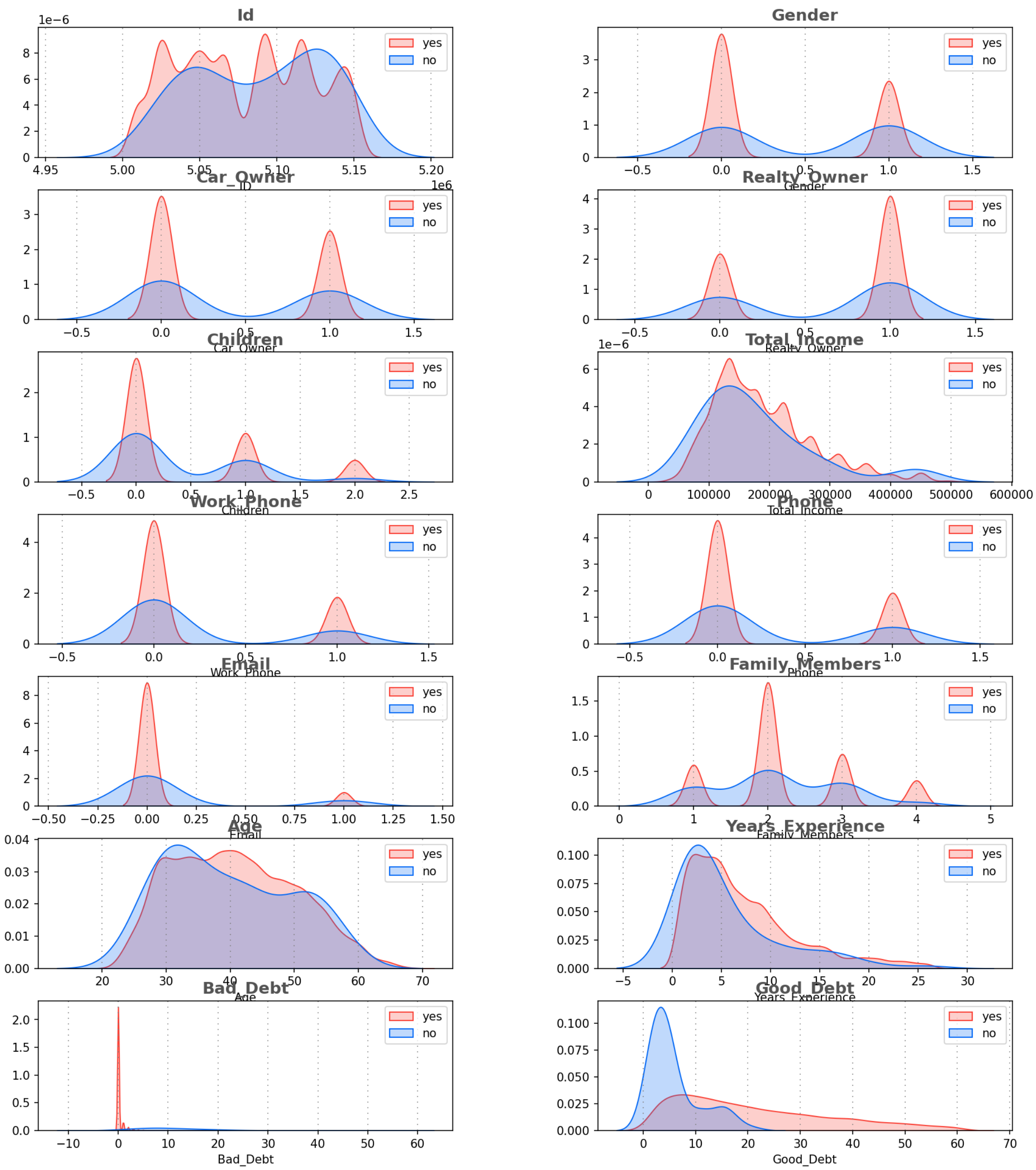
```
Number of categorical variables: 5
Number of numerical variables: 14
```

[67]:
```
#create a no group and yes group dataframes to see all the distribution of the␣
 ↪other labels corresponding to each category.
yes_gr = df[df['Status']==1]
no_gr = df[df['Status']==0]

print("group_yes' data shape: {}".format(yes_gr.shape))
print("group_no's data shape: {}".format(no_gr.shape))
```

```
group_yes' data shape: (23824, 20)
group_no's data shape: (82, 20)
```

[68]:

```
#Lets plot firs the numerical values
background_color = '#ffffff'

fig = plt.figure(figsize=(15,20), dpi=150)
fig.patch.set_facecolor(background_color) # set up background color
gs = fig.add_gridspec(8, 2)
gs.update(wspace=0.35, hspace=0.25)

# axes as a list
axes = [fig.add_subplot(gs[0,0]),
       fig.add_subplot(gs[0,1]),
       fig.add_subplot(gs[1,0]),
       fig.add_subplot(gs[1,1]),
       fig.add_subplot(gs[2,0]),
       fig.add_subplot(gs[2,1]),
       fig.add_subplot(gs[3,0]),
       fig.add_subplot(gs[3,1]),
       fig.add_subplot(gs[4,0]),
       fig.add_subplot(gs[4,1]),
       fig.add_subplot(gs[5,0]),
       fig.add_subplot(gs[5,1]),
       fig.add_subplot(gs[6,0]),
       fig.add_subplot(gs[6,1])]

#Function to create hist plots
def HistPlot(df, var, ax):
    # create histograms
    sns.kdeplot(yes_gr[var], ax=ax, color='#FA4035', shade=True, label='yes')
    sns.kdeplot(no_gr[var], ax=ax, color='#0569f5', shade=True, label='no')
    ax.grid(which='major', color='gray', linestyle=':', axis='x', zorder=0,␣
 ↪dashes=(1,5))
    ax.set_title(f'{var}'.title(), fontsize=14, fontweight='bold',
                fontfamily='DejaVu Sans', color='#535353', loc='center')
    ax.legend(loc=1)
    ax.set_ylabel('')


for ax, var in zip(axes, num_vars):
    HistPlot(df, var, ax)
```




```
[69]: # visuzalitions libraries
import matplotlib.ticker as mtick
import matplotlib.gridspec as grid_spec
def barPerc(df, variable, ax):
    """
    source: https://stackoverflow.com/a/67076347/4852724

    barPerc(): Add percentage for hues to bar plots
    args:
```

```
        df: pandas dataframe
        xVar: (string) X variable
        ax: Axes object (for Seaborn Countplot/Bar plot or
                             pandas bar plot)
    """

    # 1. How many X categories
    ## check for NaN and remove
    numX = len([x for x in df[variable].unique() if x==x])

    # 2. The bars are created in hue order, organize them
    bars = ax.patches
    ## 2a. For each X variable
    for ind in range(numX):
        ## 2b. Get hue bar
        ##     ex. 8 X categories, 4 hues =>
        ##     [0, 8, 16, 24] are hue bars for 1st X category
        hueBars = bars[ind:][::numX]
        ## 2c. Get the total height (for percentages)
        total = sum([x.get_height() for x in hueBars])

        # 3. Print the percentage on the bars
        for bar in hueBars:
            ax.text(bar.get_x() + bar.get_width()/2.,
                    bar.get_height() - 0.1 * bar.get_height(),
                    f"{bar.get_height()/total:.2%}",
                    ha='center', va='top', color='black')

def GrpSubplot(df, variable, ax, axis=None, ticklabels=None):
    """ Create subplots based on X variables"""

    df.groupby([variable, target]).size().unstack(target).apply(lambda x: x*100/
 ↪x.sum(), axis=axis).plot.bar(rot=0,
                                                                                 ␣
 ↪                             width=0.9,
                                                                                 ␣
 ↪                              alpha=0.65,
                                                                                 ␣
 ↪                             color=['#67a9cf','#FA4035'],
                                                                                 ␣
 ↪                             ax=ax);
    ax.set_title(f'{variable}'.title(), fontsize=14, fontweight='bold',␣
 ↪fontfamily='serif', color='#323232', loc='left')
    ax.grid(color='gray', linestyle=':', axis='y', zorder=0, dashes=(1,5))
    ax.set_xticklabels(ticklabels, rotation=90)
    ax.yaxis.set_major_formatter(mtick.PercentFormatter())
```

```
    ax.yaxis.set_major_locator(mtick.MultipleLocator(10))
    ax.legend(loc=1)
    ax.set_xlabel('')

    barPerc(df, variable, ax)
```

[70]:
```
df['Education_Type'].unique()

#Income_Type          Education_Type         Family_Status          Housing_Type          .
 ↪..          Work_Phone          Phone          Email          Job_Title
```

[70]:
```
array(['Secondary / secondary special', 'Higher education',
       'Incomplete higher', 'Lower secondary', 'Academic degree'],
      dtype=object)
```

[71]:
```
df['Family_Status'].unique()
```

[71]:
```
array(['Married', 'Single / not married', 'Civil marriage', 'Separated',
       'Widow'], dtype=object)
```

[72]:
```
df['Housing_Type'].unique()
```

[72]:
```
array(['House / apartment', 'Rented apartment', 'Municipal apartment',
       'With parents', 'Co-op apartment', 'Office apartment'],
      dtype=object)
```

[73]:
```
df['Income_Type'].unique()
```

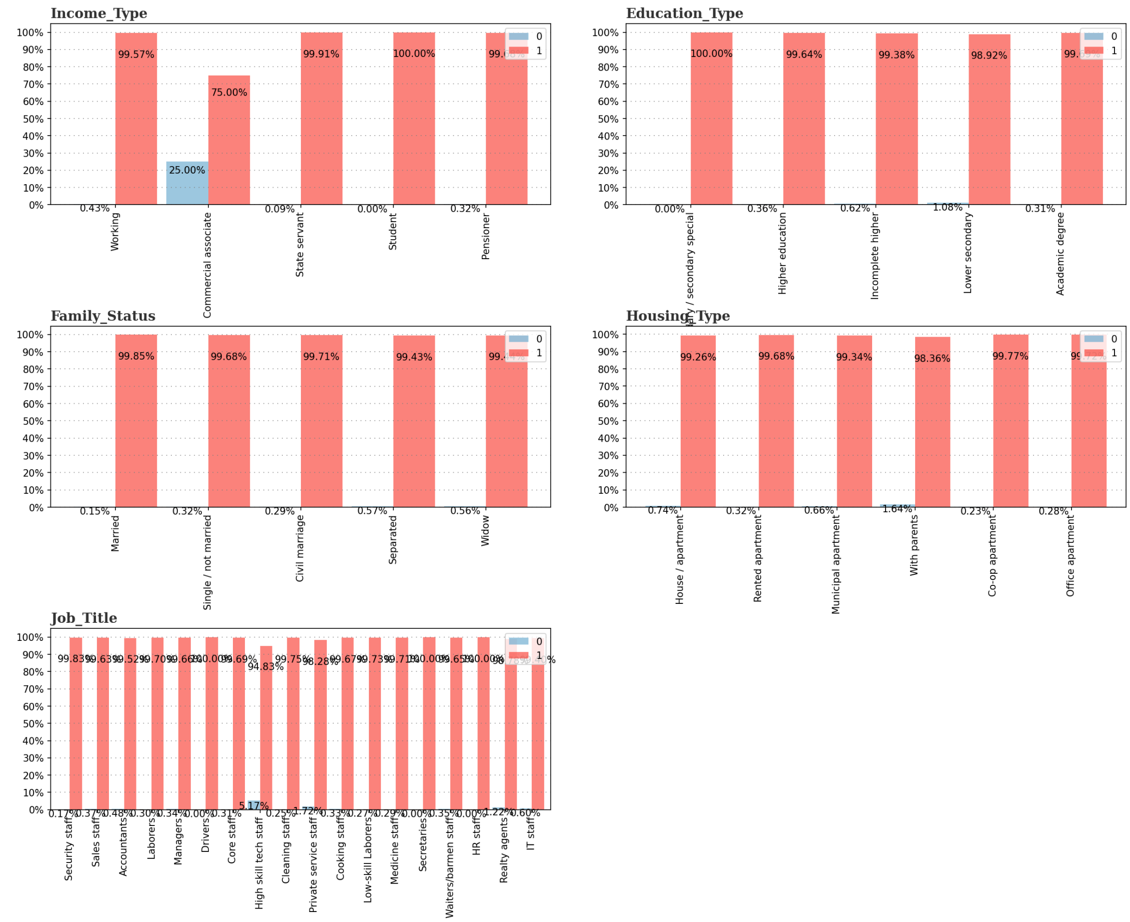
[73]:
```
array(['Working', 'Commercial associate', 'State servant', 'Student',
       'Pensioner'], dtype=object)
```

[74]:
```
df['Job_Title'].unique()
```

[74]:
```
array(['Security staff', 'Sales staff', 'Accountants', 'Laborers',
       'Managers', 'Drivers', 'Core staff', 'High skill tech staff',
       'Cleaning staff', 'Private service staff', 'Cooking staff',
       'Low-skill Laborers', 'Medicine staff', 'Secretaries',
       'Waiters/barmen staff', 'HR staff', 'Realty agents', 'IT staff'],
      dtype=object)
```

[75]:
```
fig = plt.figure(figsize=(20, 15), dpi=150)
fig.patch.set_facecolor(background_color)
gs = fig.add_gridspec(3, 2)
gs.update(wspace=0.15, hspace=0.67)

# axes as a list
axes = [fig.add_subplot(gs[0,0]),
```

```
       fig.add_subplot(gs[0,1]),
       fig.add_subplot(gs[1,0]),
       fig.add_subplot(gs[1,1]),
       fig.add_subplot(gs[2,0])]
#Income_Type        Education_Type        Family_Status        Housing_Type        .
↪..         Work_Phone        Phone        Email        Job_Title
# ticklabels as list enterend manually, time consuption to build a function for␣
↪this.
tlabs = [['Working', 'Commercial associate', 'State servant', 'Student',
       'Pensioner']] + [['Secondary / secondary special', 'Higher education',
       'Incomplete higher', 'Lower secondary', 'Academic degree']] +␣
↪[['Married', 'Single / not married', 'Civil marriage', 'Separated',
       'Widow']] + [['House / apartment', 'Rented apartment', 'Municipal␣
↪apartment',
       'With parents', 'Co-op apartment', 'Office apartment']] + [['Security␣
↪staff', 'Sales staff', 'Accountants', 'Laborers',
       'Managers', 'Drivers', 'Core staff', 'High skill tech staff',
       'Cleaning staff', 'Private service staff', 'Cooking staff',
       'Low-skill Laborers', 'Medicine staff', 'Secretaries',
       'Waiters/barmen staff', 'HR staff', 'Realty agents', 'IT staff']]*2

target = 'Status'
for ax, variable, ticklabels in zip(axes, catg_vars, tlabs):
    GrpSubplot(df, variable, ax, axis=1, ticklabels=ticklabels)
```



[76]:
```
# Now I will convert all values that are string into categories and give them
↪numerical values.  Income_Type, Education_Type, Family_Status, Housing_Type,
↪Job_Title
df['Income_Type'] = df['Income_Type'].astype('category')
df['Income_Type'] = df['Income_Type'].cat.codes
df['Education_Type'] = df['Education_Type'].astype('category')
df['Education_Type'] = df['Education_Type'].cat.codes
df['Family_Status'] = df['Family_Status'].astype('category')
df['Family_Status'] = df['Family_Status'].cat.codes
df['Housing_Type'] = df['Housing_Type'].astype('category')
df['Housing_Type'] = df['Housing_Type'].cat.codes
df['Job_Title'] = df['Job_Title'].astype('category')
df['Job_Title'] = df['Job_Title'].cat.codes
```

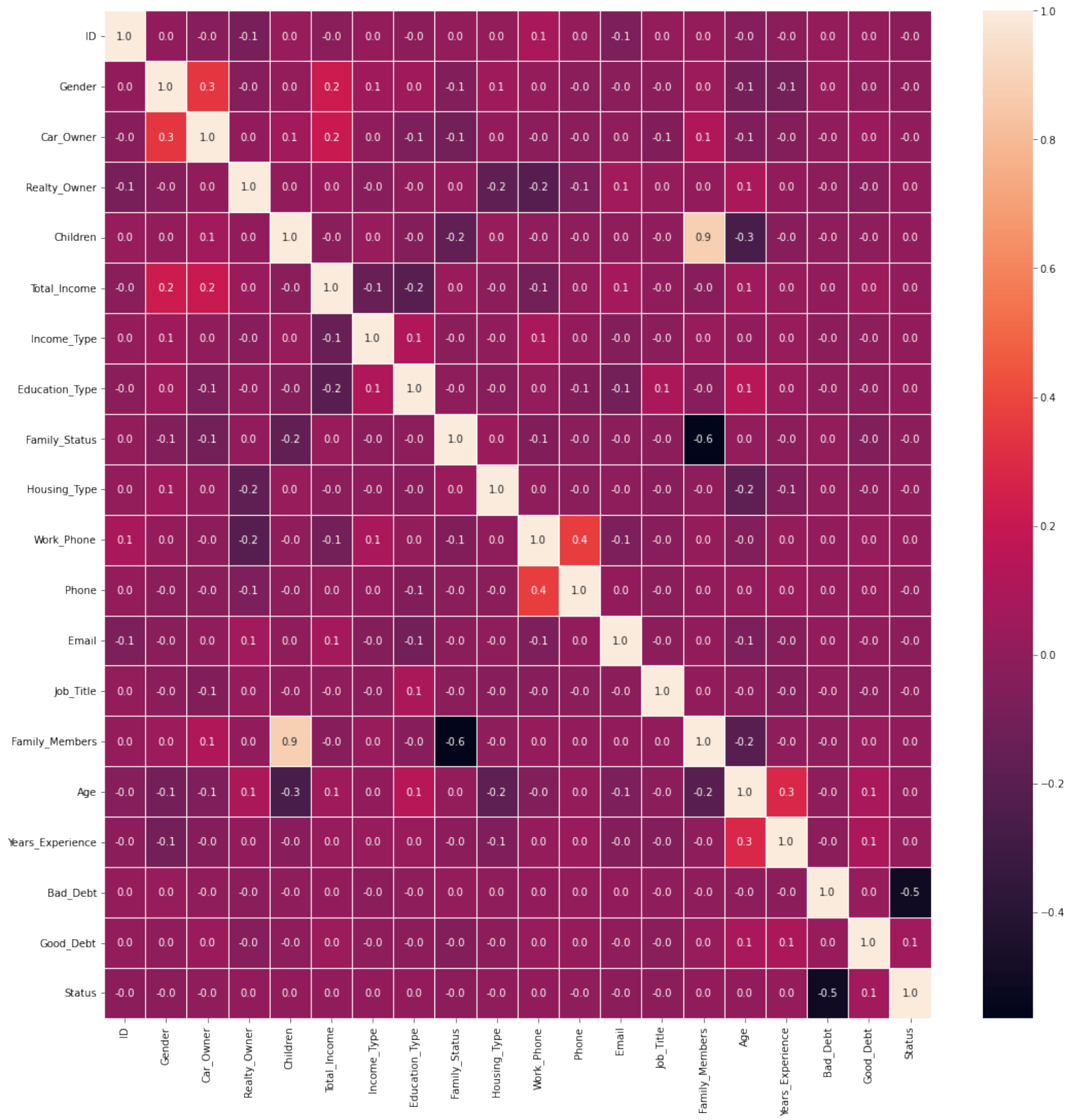
[77]:
```
df.head()
```

[77]:
```
        ID  Gender  Car_Owner  Realty_Owner  Children  Total_Income  \
0  5008806       1          1             1         0     112500.00
```

```
1  5008808       0           0             1         0     270000.00
2  5008809       0           0             1         0     270000.00
3  5008810       0           0             1         0     270000.00
4  5008811       0           0             1         0     270000.00

   Income_Type  Education_Type  Family_Status  Housing_Type  Work_Phone  \
0            4               4              1             1           0
1            0               4              3             1           0
2            0               4              3             1           0
3            0               4              3             1           0
4            0               4              3             1           0

   Phone  Email  Job_Title  Family_Members   Age  Years_Experience  Bad_Debt  \
0      0      0         16               2 59.00              4.00         0
1      1      1         14               1 53.00              9.00         0
2      1      1         14               1 53.00              9.00         0
3      1      1         14               1 53.00              9.00         0
4      1      1         14               1 53.00              9.00         0

   Good_Debt  Status
0         30       1
1          5       1
2          5       1
3         27       1
4         39       1
```

[78]:
```
#correlation map to see if we have correlating features
f,ax = plt.subplots(figsize=(18, 18))
sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

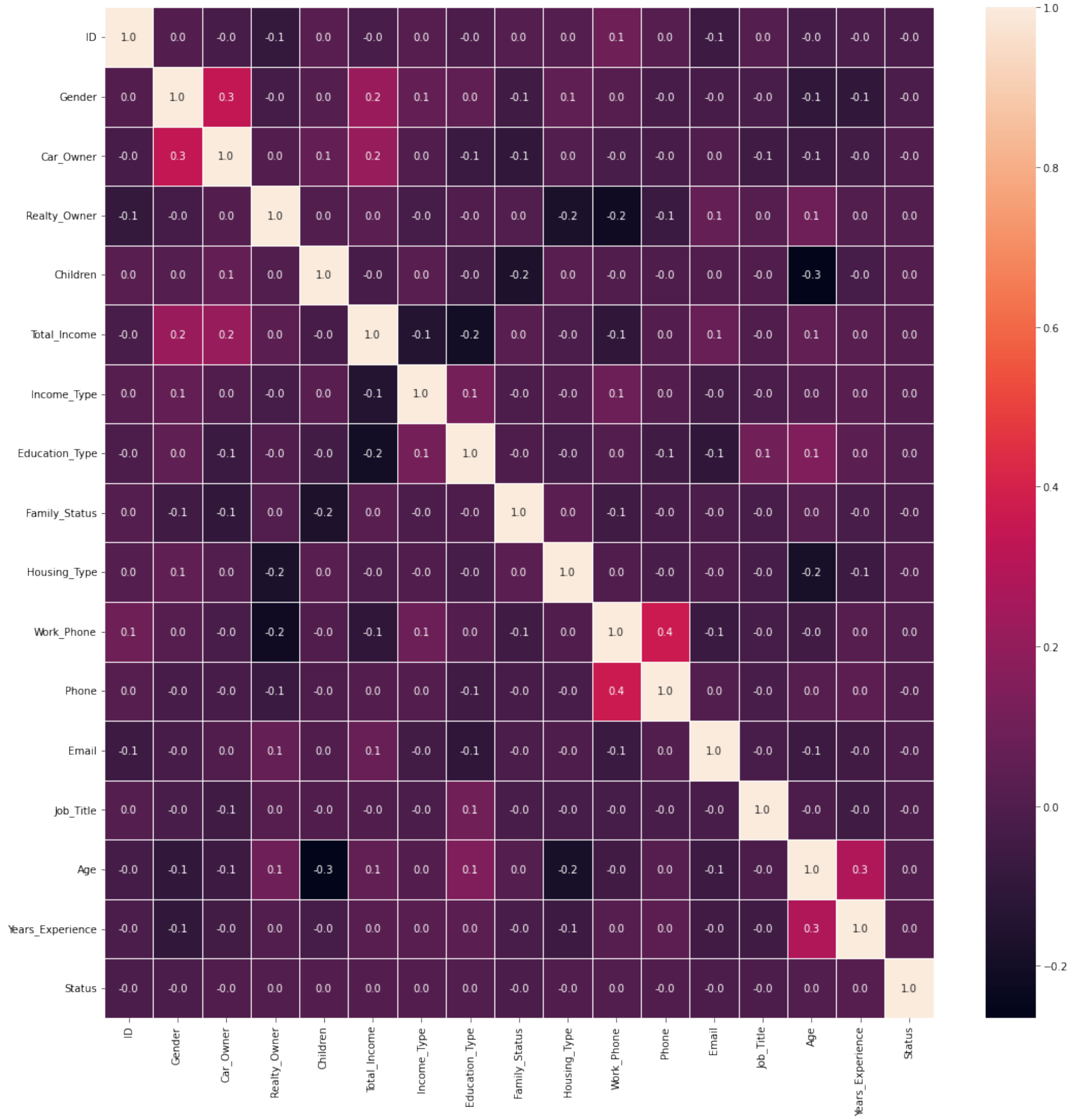
[78]: <AxesSubplot:>



[79]:
```
# From the heatmap we can see that family members and children are correlating.␣
↪so we will use only children since family members have also a bit of␣
↪correlation with family status.
df=df.drop(columns=['Family_Members','Good_Debt','Bad_Debt'])
# I will drop also the good debt and bad debt columns since I used those to get␣
↪the status results it is not sane keeping them.
```

[80]:
```
#correlation map to see if we have correlating features

f,ax = plt.subplots(figsize=(18, 18))
sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

[80]: <AxesSubplot:>



12 4- Cross-Validation

[81]:
```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, confusion_matrix, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
import itertools
#remove Status and ID for X.
```

```
X = df.drop(columns=['Status','ID'])
y = df.Status

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.
↪3,random_state=42)
```

[82]:
```
!pip install imblearn
```

```
Requirement already satisfied: imblearn in /opt/conda/lib/python3.9/site-
packages (0.0)
Requirement already satisfied: imbalanced-learn in
/opt/conda/lib/python3.9/site-packages (from imblearn) (0.9.0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.9/site-
packages (from imbalanced-learn->imblearn) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in /opt/conda/lib/python3.9/site-
packages (from imbalanced-learn->imblearn) (1.7.3)
Requirement already satisfied: numpy>=1.14.6 in /opt/conda/lib/python3.9/site-
packages (from imbalanced-learn->imblearn) (1.20.3)
Requirement already satisfied: scikit-learn>=1.0.1 in
/opt/conda/lib/python3.9/site-packages (from imbalanced-learn->imblearn) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from imbalanced-learn->imblearn) (3.0.0)
```

13 5- Traininng and testing

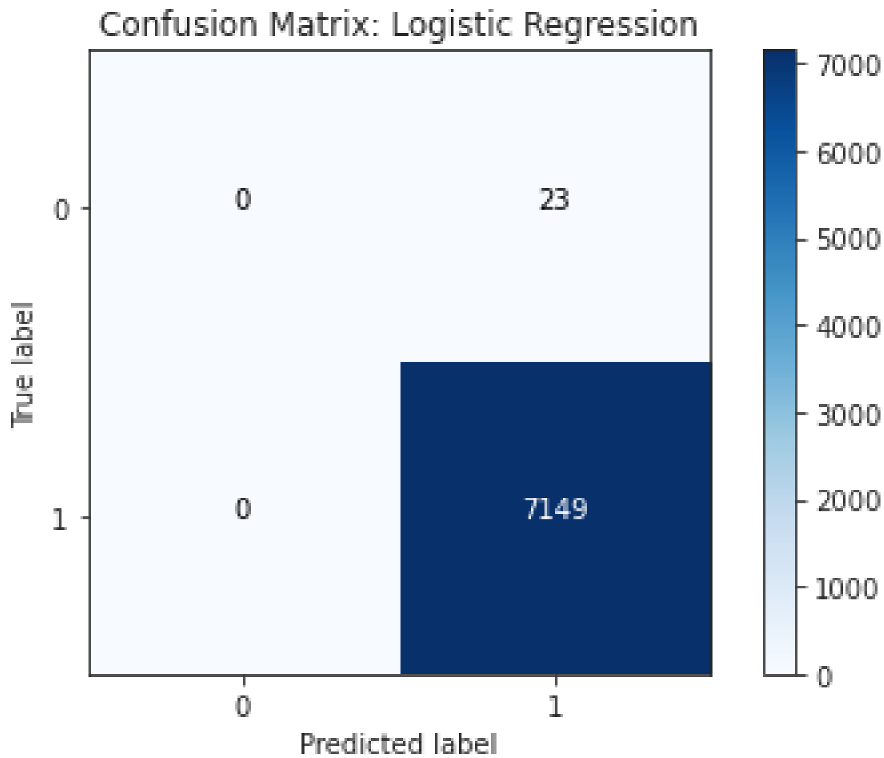
[83]:
```
#Just the ConfusionMatrix visualization taken from another source.
def vis_conf_mat(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
```

```
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

14 6.1- Logistic Regression

[84]:
```
#for simple cross-validation
model = LogisticRegression(C=0.8, penalty='l2', random_state=0, solver='lbfgs')
model.fit(X_train, y_train)
y_predict = model.predict(X_test)

print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
print(pd.DataFrame(confusion_matrix(y_test,y_predict)))

sns.set_style('ticks')
class_names = ['0','1']

vis_conf_mat(confusion_matrix(y_test,y_predict),
                      classes= class_names, normalize = False,
                      title='Confusion Matrix: Logistic Regression')
```

```
Accuracy Score is 0.99679
   0     1
0  0    23
1  0  7149
[[   0   23]
 [   0 7149]]
```



[85]:
```
f1_score(y_test, y_predict, labels=None, pos_label=1, average='binary',␣
 ↪sample_weight=None, zero_division='warn')
```

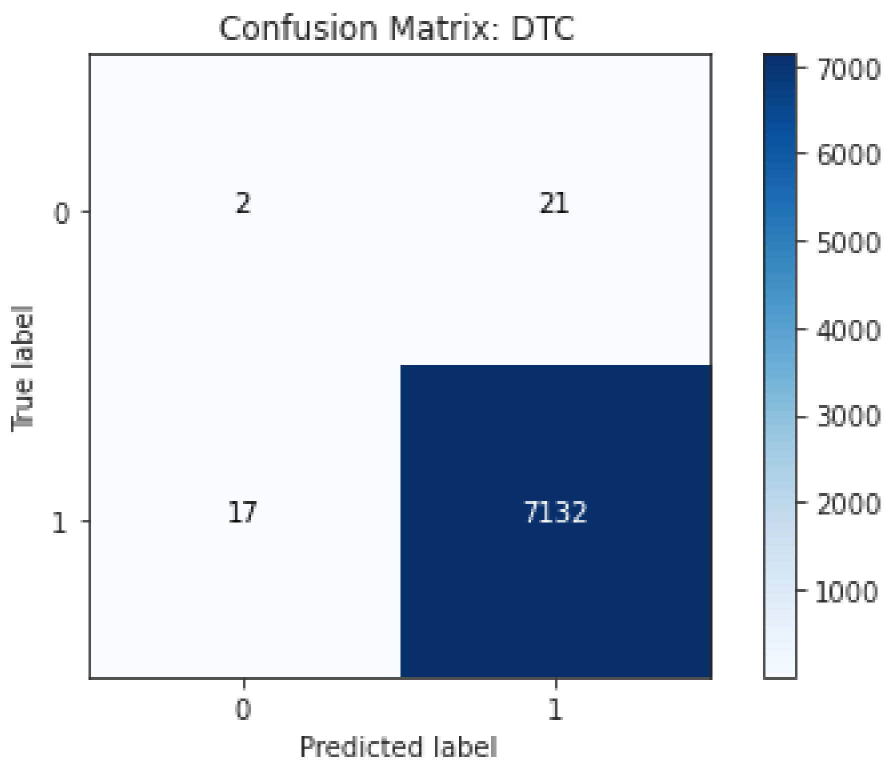
[85]: 0.9983939669017526

15 6.2- DecisionTreeClassifier

[86]:
```
model2 = DecisionTreeClassifier(max_depth=12, min_samples_split=8,␣
 ↪random_state=1024)
model2.fit(X_train, y_train)
y_predict = model2.predict(X_test)

print('Accuracy Score is {:.5}'.format(accuracy_score(y_test, y_predict)))
print(pd.DataFrame(confusion_matrix(y_test,y_predict)))

vis_conf_mat(confusion_matrix(y_test,y_predict),
                      classes=class_names, normalize = False,
                      title='Confusion Matrix: DTC')
```

```
Accuracy Score is 0.9947
    0     1
0   2    21
```

```
1   17   7132
[[    2    21]
 [   17 7132]]
```



[87]:
```
f1_score(y_test, y_predict, labels=None, pos_label=1, average='binary',␣
↪sample_weight=None, zero_division='warn')
```

[87]:
```
0.9973430289470004
```

16 Since my models are overfitting I am trying a SMOTE oversampling technique

17 EXTRA This is extra and not included in the Coursework.

[88]:
```
from imblearn.over_sampling import SMOTE
print("Before OverSampling, counts of label '1': {}".format(sum(y_train==1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train==0)))

sm = SMOTE(random_state=2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())
```

```
print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.
 ↪shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
```

```
Before OverSampling, counts of label '1': 16675
Before OverSampling, counts of label '0': 59

After OverSampling, the shape of train_X: (33350, 15)
After OverSampling, the shape of train_y: (33350,)

After OverSampling, counts of label '1': 16675
After OverSampling, counts of label '0': 16675
```

[89]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, precision_recall_curve, auc,␣
 ↪roc_auc_score, roc_curve, recall_score, classification_report

parameters = {
    'C': np.linspace(1, 10, 10)
             }
lr = LogisticRegression()
clf = GridSearchCV(lr, parameters, cv=5, verbose=5, n_jobs=3)
clf.fit(X_train_res, y_train_res.ravel())
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
```

[89]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(), n_jobs=3,
             param_grid={'C': array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,
9., 10.])},
             verbose=5)
```

[90]:
```
clf.best_params_
```

[90]:
```
{'C': 1.0}
```

[91]:
```
lr1 = LogisticRegression(C=1,penalty='none', verbose=5)
lr1.fit(X_train_res, y_train_res.ravel())
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

RUNNING THE L-BFGS-B CODE

           * * *
```

```
Machine precision = 2.220D-16
 N =           16     M =           10

 L =  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00

X0 =  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00

 U =  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.31165D+04    |proj g|=  4.22682D+07


ITERATION     1

---------------- CAUCHY entered-------------------
 There are            0   breakpoints

 GCP found in this segment
Piece      1 --f1, f2 at start point  -1.7866D+15  1.7866D+15
Distance to the stationary point =   1.0000D+00
Cauchy X =
      7.1150D+02  1.5780D+03  2.3265D+03  2.8925D+03  4.2268D+07  5.3620D+03
      4.5970D+03 -5.5500D+02  5.0250D+02  1.5285D+03  1.0430D+03  6.3450D+02
     -2.1510D+03  5.9517D+03  1.4179D+04  0.0000D+00

---------------- exit CAUCHY----------------------

          16  variables are free at GCP            1

 This problem is unconstrained.

 LINE SEARCH           3  times; norm of step =    1.1688637189698825E-007

At iterate    1    f=  2.31139D+04    |proj g|=  1.54261D+06

 X =  1.9675D-12  4.3637D-12  6.4336D-12  7.9988D-12  1.1689D-07  1.4828D-11
      1.2712D-11 -1.5348D-12  1.3896D-12  4.2268D-12  2.8843D-12  1.7546D-12
     -5.9483D-12  1.6459D-11  3.9211D-11  0.0000D+00

 G = -6.3642D+02 -1.5115D+03 -2.2339D+03 -2.8381D+03 -1.5426D+06 -4.9613D+03
     -4.1130D+03  8.0508D+02 -2.6897D+02 -1.4989D+03 -1.0032D+03 -6.2079D+02
```

```
      3.5246D+03  1.2718D+03 -1.3057D+04  1.8045D+02


ITERATION     2

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    4.4277149121571538E-009

At iterate    2    f=  2.31139D+04    |proj g|=  1.30146D+04

 X =  3.8605D-12  8.8598D-12  1.3079D-11  1.6441D-11  1.2131D-07  2.9585D-11
      2.4946D-11 -3.9308D-12  2.1888D-12  8.6857D-12  5.8686D-12  3.6013D-12
     -1.6439D-11  1.2647D-11  7.8049D-11 -5.3754D-13

 G = -6.3357D+02 -1.5089D+03 -2.2304D+03 -2.8360D+03 -2.9699D+02 -4.9462D+03
     -4.0947D+03  8.1455D+02 -2.6013D+02 -1.4978D+03 -1.0017D+03 -6.2027D+02
      3.5767D+03  1.5454D+03 -1.3015D+04  1.8729D+02


ITERATION     3

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           4  times; norm of step =    1.5362792122841476E-008

At iterate    3    f=  2.31139D+04    |proj g|=  1.57337D+05

 X =  6.2432D-10  1.4866D-09  2.1973D-09  2.7938D-09  1.2177D-07  4.8734D-09
      4.0349D-09 -8.0162D-10  2.5693D-10  1.4755D-09  9.8687D-10  6.1104D-10
     -3.5191D-09 -1.5007D-09  1.2823D-08 -1.8395D-10

 G = -6.3328D+02 -1.5087D+03 -2.2300D+03 -2.8358D+03  1.5734D+05 -4.9446D+03
     -4.0928D+03  8.1552D+02 -2.5922D+02 -1.4977D+03 -1.0016D+03 -6.2022D+02
      3.5820D+03  1.5733D+03 -1.3010D+04  1.8799D+02


ITERATION     4

----------------SUBSM entered-----------------
```

```
----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    8.8365558097578073E-008

At iterate    4    f=  2.31139D+04    |proj g|=  5.32118D+05

 X =  4.1944D-09  9.9893D-09  1.4765D-08  1.8775D-08  1.2284D-07  3.2744D-08
      2.7108D-08 -5.3915D-09  1.7227D-09  9.9152D-09  6.6315D-09  4.1062D-09
     -2.3673D-08 -1.0209D-08  8.6159D-08 -1.2393D-09

 G = -6.3259D+02 -1.5081D+03 -2.2291D+03 -2.8353D+03  5.3212D+05 -4.9409D+03
     -4.0883D+03  8.1782D+02 -2.5707D+02 -1.4974D+03 -1.0012D+03 -6.2010D+02
      3.5946D+03  1.6398D+03 -1.3000D+04  1.8965D+02


ITERATION     5

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.3456306291274544E-007

At iterate    5    f=  2.31139D+04    |proj g|=  1.28780D+06

 X =  1.7712D-08  4.2183D-08  6.2351D-08  7.9283D-08  1.2501D-07  1.3827D-07
      1.1447D-07 -2.2770D-08  7.2726D-09  4.1871D-08  2.8004D-08  1.7340D-08
     -9.9983D-08 -4.3181D-08  3.6383D-07 -5.2352D-09

 G = -6.3120D+02 -1.5068D+03 -2.2274D+03 -2.8343D+03  1.2878D+06 -4.9335D+03
     -4.0793D+03  8.2246D+02 -2.5274D+02 -1.4968D+03 -1.0005D+03 -6.1984D+02
      3.6201D+03  1.7739D+03 -1.2979D+04  1.9300D+02


ITERATION     6

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    8.9083116877568545E-007

At iterate    6    f=  2.31139D+04    |proj g|=  2.41823D+06

 X =  5.3705D-08  1.2791D-07  1.8906D-07  2.4040D-07  1.2825D-07  4.1926D-07
      3.4709D-07 -6.9046D-08  2.2050D-08  1.2696D-07  8.4913D-08  5.2578D-08
```

```
     -3.0317D-07 -1.3098D-07  1.1032D-06 -1.5875D-08

 G = -6.2911D+02 -1.5050D+03 -2.2249D+03 -2.8328D+03  2.4182D+06 -4.9223D+03
     -4.0659D+03  8.2941D+02 -2.4625D+02 -1.4960D+03 -9.9937D+02 -6.1946D+02
      3.6582D+03  1.9745D+03 -1.2948D+04  1.9801D+02


ITERATION     7

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.5279011834938847E-006

At iterate    7    f=  2.31139D+04    |proj g|=  4.30422D+06

 X =  1.5584D-07  3.7116D-07  5.4862D-07  6.9760D-07  1.3365D-07  1.2166D-06
      1.0072D-06 -2.0036D-07  6.3985D-08  3.6842D-07  2.4640D-07  1.5257D-07
     -8.7977D-07 -3.8011D-07  3.2013D-06 -4.6068D-08

 G = -6.2564D+02 -1.5019D+03 -2.2206D+03 -2.8303D+03  4.3042D+06 -4.9037D+03
     -4.0434D+03  8.4099D+02 -2.3542D+02 -1.4946D+03 -9.9753D+02 -6.1883D+02
      3.7218D+03  2.3092D+03 -1.2895D+04  2.0637D+02


ITERATION     8

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    6.7501273852657410E-006

At iterate    8    f=  2.31138D+04    |proj g|=  7.32020D+06

 X =  4.2858D-07  1.0207D-06  1.5087D-06  1.9184D-06  1.4228D-07  3.3458D-06
      2.7698D-06 -5.5100D-07  1.7596D-07  1.0132D-06  6.7762D-07  4.1958D-07
     -2.4194D-06 -1.0454D-06  8.8037D-06 -1.2669D-07

 G = -6.2008D+02 -1.4970D+03 -2.2137D+03 -2.8262D+03  7.3202D+06 -4.8739D+03
     -4.0075D+03  8.5951D+02 -2.1809D+02 -1.4924D+03 -9.9458D+02 -6.1781D+02
      3.8234D+03  2.8448D+03 -1.2811D+04  2.1975D+02


ITERATION     9
```

```
----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.8075492157081601E-005

At iterate    9    f=  2.31137D+04    |proj g|=  1.22204D+07

 X =  1.1589D-06  2.7601D-06  4.0797D-06  5.1876D-06  1.5627D-07  9.0473D-06
      7.4898D-06 -1.4900D-06  4.7581D-07  2.7397D-06  1.8323D-06  1.1346D-06
     -6.5423D-06 -2.8268D-06  2.3806D-05 -3.4258D-07

 G = -6.1105D+02 -1.4889D+03 -2.2025D+03 -2.8196D+03  1.2220D+07 -4.8253D+03
     -3.9489D+03  8.8960D+02 -1.8990D+02 -1.4888D+03 -9.8979D+02 -6.1616D+02
      3.9883D+03  3.7157D+03 -1.2673D+04  2.4151D+02


ITERATION    10

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    4.7764259490236209E-005

At iterate   10    f=  2.31133D+04    |proj g|=  2.01293D+07

 X =  3.0888D-06  7.3564D-06  1.0873D-05  1.3826D-05  1.7876D-07  2.4114D-05
      1.9962D-05 -3.9711D-06  1.2682D-06  7.3020D-06  4.8837D-06  3.0240D-06
     -1.7437D-05 -7.5342D-06  6.3449D-05 -9.1308D-07

 G = -5.9648D+02 -1.4759D+03 -2.1843D+03 -2.8088D+03  2.0129D+07 -4.7466D+03
     -3.8541D+03  9.3816D+02 -1.4431D+02 -1.4830D+03 -9.8205D+02 -6.1350D+02
      4.2542D+03  5.1229D+03 -1.2446D+04  2.7666D+02


ITERATION    11

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.2586733304802177E-004
```

```
At iterate   11    f=  2.31123D+04    |proj g|=  3.29085D+07

 X =  8.1744D-06  1.9469D-05  2.8776D-05  3.6591D-05  2.1489D-07  6.3816D-05
      5.2830D-05 -1.0509D-05  3.3562D-06  1.9324D-05  1.2924D-05  8.0028D-06
     -4.6147D-05 -1.9939D-05  1.6792D-04 -2.4164D-06

 G = -5.7294D+02 -1.4548D+03 -2.1549D+03 -2.7912D+03  3.2909D+07 -4.6184D+03
     -3.7002D+03  1.0166D+03 -7.0394D+01 -1.4735D+03 -9.6951D+02 -6.0920D+02
      4.6829D+03  7.4011D+03 -1.2070D+04  3.3357D+02


ITERATION    12

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.2999506718716786E-004

At iterate   12    f=  2.31097D+04    |proj g|=  5.34724D+07

 X =  2.1508D-05  5.1224D-05  7.5713D-05  9.6274D-05  2.7247D-07  1.6791D-04
      1.3900D-04 -2.7652D-05  8.8304D-06  5.0845D-05  3.4006D-05  2.1056D-05
     -1.2142D-04 -5.2462D-05  4.4180D-04 -6.3579D-06

 G = -5.3509D+02 -1.4205D+03 -2.1070D+03 -2.7624D+03  5.3472D+07 -4.4095D+03
     -3.4502D+03  1.1429D+03  4.9192D+01 -1.4580D+03 -9.4927D+02 -6.0228D+02
      5.3703D+03  1.1079D+04 -1.1443D+04  4.2543D+02


ITERATION    13

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    8.6046149006244362E-004

At iterate   13    f=  2.31029D+04    |proj g|=  8.62983D+07

 X =  5.6274D-05  1.3403D-04  1.9810D-04  2.5190D-04  3.6294D-07  4.3932D-04
      3.6369D-04 -7.2349D-05  2.3104D-05  1.3303D-04  8.8975D-05  5.5093D-05
     -3.1768D-04 -1.3726D-04  1.1560D-03 -1.6635D-05

 G = -4.7474D+02 -1.3652D+03 -2.0296D+03 -2.7149D+03  8.6298D+07 -4.0696D+03
     -3.0457D+03  1.3444D+03  2.4178D+02 -1.4327D+03 -9.1681D+02 -5.9123D+02
```

```
      6.4615D+03  1.6980D+04 -1.0380D+04  5.7282D+02


ITERATION    14

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.2172296756586439E-003

At iterate   14    f=  2.30856D+04    |proj g|=  1.37552D+08

 X =  1.4586D-04  3.4739D-04  5.1347D-04  6.5291D-04  5.0048D-07  1.1387D-03
      9.4267D-04 -1.8753D-04  5.9886D-05  3.4482D-04  2.3062D-04  1.4280D-04
     -8.2342D-04 -3.5578D-04  2.9962D-03 -4.3118D-05

 G = -3.8067D+02 -1.2771D+03 -1.9059D+03 -2.6372D+03  1.3755D+08 -3.5215D+03
     -2.3995D+03  1.6590D+03  5.4692D+02 -1.3918D+03 -8.6570D+02 -5.7398D+02
      8.1492D+03  2.6275D+04 -8.5593D+03  8.0495D+02


ITERATION    15

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    5.5548891178574082E-003

At iterate   15    f=  2.30425D+04    |proj g|=  2.13061D+08

 X =  3.7030D-04  8.8193D-04  1.3036D-03  1.6576D-03  6.9361D-07  2.8909D-03
      2.3932D-03 -4.7608D-04  1.5204D-04  8.7540D-04  5.8548D-04  3.6253D-04
     -2.0905D-03 -9.0325D-04  7.6066D-03 -1.0947D-04

 G = -2.4249D+02 -1.1428D+03 -1.7164D+03 -2.5131D+03  2.1306D+08 -2.6692D+03
     -1.4094D+03  2.1226D+03  1.0080D+03 -1.3277D+03 -7.8932D+02 -5.4857D+02
      1.0594D+04  4.0178D+04 -5.4617D+03  1.1521D+03


ITERATION    16

----------------SUBSM entered-----------------
```

```
----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.2977148019832284E-002

At iterate   16    f=  2.29435D+04    |proj g|=  3.07751D+08

 X =  8.9464D-04  2.1307D-03  3.1494D-03  4.0046D-03  9.1160D-07  6.9842D-03
      5.7819D-03 -1.1502D-03  3.6731D-04  2.1149D-03  1.4145D-03  8.7586D-04
     -5.0505D-03 -2.1822D-03  1.8377D-02 -2.6446D-04

 G = -7.0193D+01 -9.6278D+02 -1.4601D+03 -2.3332D+03  3.0775D+08 -1.4857D+03
     -7.0916D+01  2.7033D+03  1.6160D+03 -1.2378D+03 -6.9088D+02 -5.1675D+02
      1.3548D+04  5.8126D+04 -5.1618D+02  1.6004D+03


ITERATION    17

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.5747134264788408E-002

At iterate   17    f=  2.27540D+04    |proj g|=  3.77268D+08

 X =  1.9349D-03  4.6083D-03  6.8116D-03  8.6613D-03  1.0086D-06  1.5106D-02
      1.2505D-02 -2.4877D-03  7.9443D-04  4.5742D-03  3.0593D-03  1.8943D-03
     -1.0923D-02 -4.7197D-03  3.9747D-02 -5.7199D-04

 G =  5.4008D+01 -7.9975D+02 -1.2244D+03 -2.1371D+03  3.7727D+08 -3.2089D+02
      1.1578D+03  3.1247D+03  2.1385D+03 -1.1470D+03 -6.1230D+02 -4.9353D+02
      1.5401D+04  7.2507D+04  5.8391D+03  1.9612D+03


ITERATION    18

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.6147542417118275E-002

At iterate   18    f=  2.25071D+04    |proj g|=  3.26551D+08

 X =  3.3955D-03  8.0868D-03  1.1953D-02  1.5199D-02  7.2474D-07  2.6508D-02
      2.1944D-02 -4.3654D-03  1.3941D-03  8.0269D-03  5.3685D-03  3.3242D-03
```

```
    -1.9168D-02 -8.2822D-03  6.9748D-02 -1.0037D-03

 G = -4.3478D+01 -8.1430D+02 -1.2385D+03 -2.0664D+03  3.2655D+08 -1.8489D+02
      1.0748D+03  2.7949D+03  2.0098D+03 -1.1306D+03 -6.4926D+02 -5.1080D+02
      1.2963D+04  6.5840D+04  1.0256D+04  1.8006D+03


ITERATION    19

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.4716490820984583E-002

At iterate   19    f=  2.23553D+04    |proj g|=  1.53027D+08

 X =  4.3941D-03  1.0465D-02  1.5469D-02  1.9669D-02  1.2338D-07  3.4304D-02
      2.8398D-02 -5.6493D-03  1.8041D-03  1.0388D-02  6.9475D-03  4.3019D-03
     -2.4806D-02 -1.0718D-02  9.0262D-02 -1.2989D-03

 G = -3.6504D+02 -1.0515D+03 -1.5662D+03 -2.2057D+03  1.5303D+08 -1.4700D+03
     -6.4689D+02  1.7140D+03  1.1203D+03 -1.2214D+03 -8.1086D+02 -5.6898D+02
      6.6274D+03  3.6517D+04  9.3447D+03  1.0721D+03


ITERATION    20

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.8930027746198238E-003

At iterate   20    f=  2.23232D+04    |proj g|=  3.55989D+07

 X =  4.4706D-03  1.0647D-02  1.5738D-02  2.0012D-02 -2.3554D-07  3.4901D-02
      2.8893D-02 -5.7477D-03  1.8355D-03  1.0569D-02  7.0684D-03  4.3768D-03
     -2.5238D-02 -1.0905D-02  9.1834D-02 -1.3216D-03

 G = -5.8210D+02 -1.2397D+03 -1.8262D+03 -2.3539D+03  3.5599D+07 -2.5838D+03
     -2.0073D+03  9.9363D+02  4.5545D+02 -1.3033D+03 -9.2439D+02 -6.0827D+02
      2.6435D+03  1.5945D+04  6.5571D+03  5.5652D+02


ITERATION    21
```

```
----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.9279214218366186E-003

At iterate   21    f=  2.23195D+04    |proj g|=  9.44705D+05

 X =  4.3119D-03  1.0269D-02  1.5179D-02  1.9301D-02 -3.2546D-07  3.3662D-02
      2.7867D-02 -5.5436D-03  1.7703D-03  1.0193D-02  6.8175D-03  4.2214D-03
     -2.4342D-02 -1.0518D-02  8.8573D-02 -1.2746D-03

 G = -6.4612D+02 -1.3034D+03 -1.9144D+03 -2.4136D+03  9.4471D+05 -2.9851D+03
     -2.4670D+03  7.8390D+02  2.4060D+02 -1.3336D+03 -9.5908D+02 -6.1988D+02
      1.5526D+03  9.6463D+03  5.0849D+03  3.9754D+02


ITERATION    22

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.4958486473720335E-003

At iterate   22    f=  2.23193D+04    |proj g|=  7.81593D+05

 X =  4.2515D-03  1.0125D-02  1.4966D-02  1.9031D-02 -3.2517D-07  3.3190D-02
      2.7476D-02 -5.4659D-03  1.7455D-03  1.0051D-02  6.7220D-03  4.1622D-03
     -2.4001D-02 -1.0370D-02  8.7332D-02 -1.2568D-03

 G = -6.4929D+02 -1.3089D+03 -1.9222D+03 -2.4212D+03 -7.8159D+05 -3.0262D+03
     -2.5068D+03  7.7423D+02  2.2452D+02 -1.3369D+03 -9.6114D+02 -6.2047D+02
      1.5227D+03  9.2651D+03  4.8222D+03  3.8762D+02


ITERATION    23

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.3371981880801531E-004
```

```
At iterate   23    f=  2.23193D+04    |proj g|=  9.39512D+03

 X =  4.2461D-03  1.0113D-02  1.4947D-02  1.9006D-02 -3.2236D-07  3.3148D-02
      2.7442D-02 -5.4590D-03  1.7433D-03  1.0038D-02  6.7134D-03  4.1569D-03
     -2.3970D-02 -1.0357D-02  8.7221D-02 -1.2552D-03

 G = -6.4785D+02 -1.3079D+03 -1.9207D+03 -2.4206D+03 -4.0032D+03 -3.0207D+03
     -2.4993D+03  7.7907D+02  2.2842D+02 -1.3365D+03 -9.6042D+02 -6.2020D+02
      1.5514D+03  9.3951D+03  4.8232D+03  3.9085D+02


ITERATION    24

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.5831880621849152E-007

At iterate   24    f=  2.23193D+04    |proj g|=  9.39630D+03

 X =  4.2461D-03  1.0113D-02  1.4947D-02  1.9006D-02 -3.2234D-07  3.3148D-02
      2.7441D-02 -5.4590D-03  1.7433D-03  1.0038D-02  6.7134D-03  4.1569D-03
     -2.3970D-02 -1.0357D-02  8.7221D-02 -1.2552D-03

 G = -6.4784D+02 -1.3079D+03 -1.9207D+03 -2.4206D+03  2.7460D+03 -3.0207D+03
     -2.4992D+03  7.7912D+02  2.2846D+02 -1.3365D+03 -9.6042D+02 -6.2020D+02
      1.5516D+03  9.3963D+03  4.8234D+03  3.9088D+02

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   16     24     32      1     0     0   9.396D+03   2.232D+04

 X =  4.2461D-03  1.0113D-02  1.4947D-02  1.9006D-02 -3.2234D-07  3.3148D-02
      2.7441D-02 -5.4590D-03  1.7433D-03  1.0038D-02  6.7134D-03  4.1569D-03
     -2.3970D-02 -1.0357D-02  8.7221D-02 -1.2552D-03
```

```
  F =   22319.297138709000

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    6.4s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    6.4s finished
```

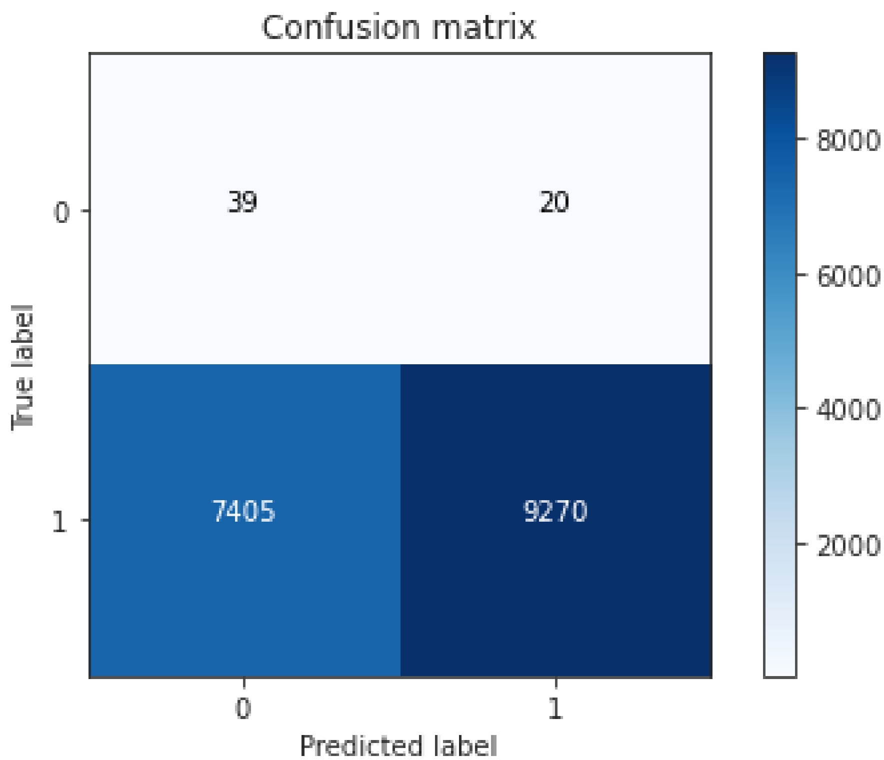
[91]:
```
LogisticRegression(C=1, penalty='none', verbose=5)
```

[92]:
```
import itertools

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=0)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        #print("Normalized confusion matrix")
    else:
        1#print('Confusion matrix, without normalization')

    #print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[93]:
```
y_train_pre = lr1.predict(X_train)

cnf_matrix_tra = confusion_matrix(y_train, y_train_pre)
```

```
print("Recall metric in the train dataset: {}%".format(100*cnf_matrix_tra[1,1]/
 ↪(cnf_matrix_tra[1,0]+cnf_matrix_tra[1,1])))


class_names = [0,1]
plt.figure()
plot_confusion_matrix(cnf_matrix_tra , classes=class_names, title='Confusion␣
 ↪matrix')
plt.show()
```

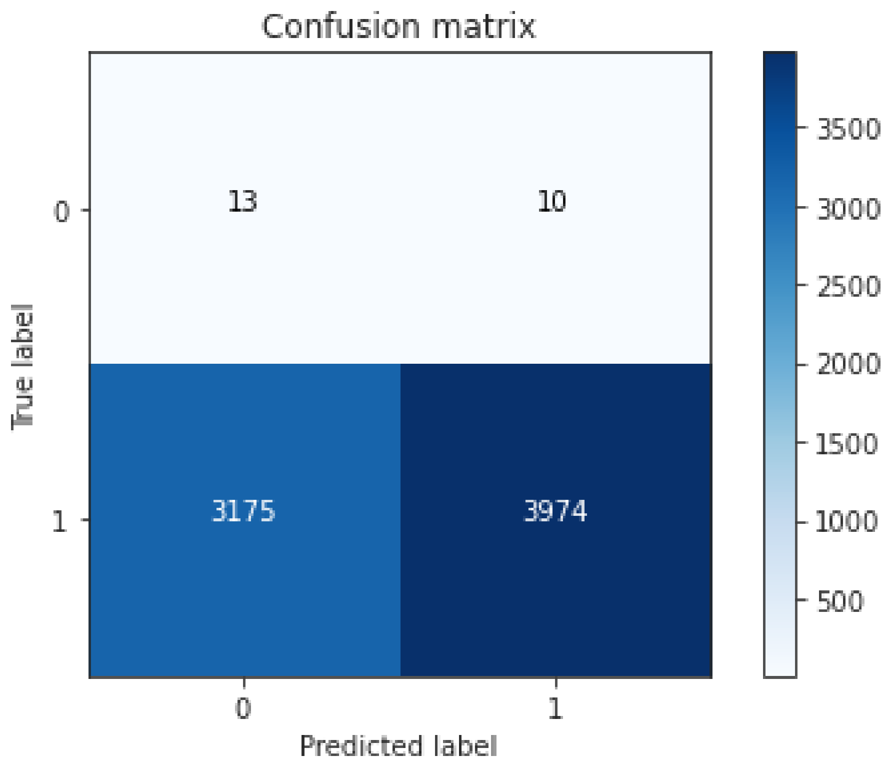
```
Recall metric in the train dataset: 55.592203898050975%
```



[94]:
```
y_pre = lr1.predict(X_test)

cnf_matrix = confusion_matrix(y_test, y_pre)

print("Recall metric in the testing dataset: {}%".format(100*cnf_matrix[1,1]/
 ↪(cnf_matrix[1,0]+cnf_matrix[1,1])))
#print("Precision metric in the testing dataset: {}%".
 ↪format(100*cnf_matrix[0,0]/(cnf_matrix[0,0]+cnf_matrix[1,0])))
# Plot confusion matrix
class_names = [0,1]
```

```
plt.figure()
plot_confusion_matrix(cnf_matrix , classes=class_names, title='Confusion␣
  ↪matrix')
plt.show()
```

Recall metric in the testing dataset: 55.5881941530284%



[95]:
```
tmp = lr1.fit(X_train_res, y_train_res.ravel())
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
 This problem is unconstrained.

RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =           16     M =           10

 L =  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
```

```
X0 =  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00

 U =  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00
      0.0000D+00  0.0000D+00  0.0000D+00  0.0000D+00

At X0         0 variables are exactly at the bounds

At iterate    0    f=  2.31165D+04    |proj g|=  4.22682D+07


ITERATION     1

---------------- CAUCHY entered-------------------
 There are            0   breakpoints

 GCP found in this segment
Piece      1 --f1, f2 at start point  -1.7866D+15  1.7866D+15
Distance to the stationary point =   1.0000D+00
Cauchy X =
      7.1150D+02  1.5780D+03  2.3265D+03  2.8925D+03  4.2268D+07  5.3620D+03
      4.5970D+03 -5.5500D+02  5.0250D+02  1.5285D+03  1.0430D+03  6.3450D+02
     -2.1510D+03  5.9517D+03  1.4179D+04  0.0000D+00

---------------- exit CAUCHY----------------------

          16  variables are free at GCP            1
 LINE SEARCH           3  times; norm of step =    1.1688637189698825E-007

At iterate    1    f=  2.31139D+04    |proj g|=  1.54261D+06

 X =  1.9675D-12  4.3637D-12  6.4336D-12  7.9988D-12  1.1689D-07  1.4828D-11
      1.2712D-11 -1.5348D-12  1.3896D-12  4.2268D-12  2.8843D-12  1.7546D-12
     -5.9483D-12  1.6459D-11  3.9211D-11  0.0000D+00

 G = -6.3642D+02 -1.5115D+03 -2.2339D+03 -2.8381D+03 -1.5426D+06 -4.9613D+03
     -4.1130D+03  8.0508D+02 -2.6897D+02 -1.4989D+03 -1.0032D+03 -6.2079D+02
      3.5246D+03  1.2718D+03 -1.3057D+04  1.8045D+02


ITERATION     2

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------
```

```
 LINE SEARCH           0  times; norm of step =    4.4277149121571538E-009

At iterate    2    f=  2.31139D+04    |proj g|=  1.30146D+04

 X =  3.8605D-12  8.8598D-12  1.3079D-11  1.6441D-11  1.2131D-07  2.9585D-11
      2.4946D-11 -3.9308D-12  2.1888D-12  8.6857D-12  5.8686D-12  3.6013D-12
     -1.6439D-11  1.2647D-11  7.8049D-11 -5.3754D-13

 G = -6.3357D+02 -1.5089D+03 -2.2304D+03 -2.8360D+03 -2.9699D+02 -4.9462D+03
     -4.0947D+03  8.1455D+02 -2.6013D+02 -1.4978D+03 -1.0017D+03 -6.2027D+02
      3.5767D+03  1.5454D+03 -1.3015D+04  1.8729D+02


ITERATION     3

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           4  times; norm of step =    1.5362792122841476E-008

At iterate    3    f=  2.31139D+04    |proj g|=  1.57337D+05

 X =  6.2432D-10  1.4866D-09  2.1973D-09  2.7938D-09  1.2177D-07  4.8734D-09
      4.0349D-09 -8.0162D-10  2.5693D-10  1.4755D-09  9.8687D-10  6.1104D-10
     -3.5191D-09 -1.5007D-09  1.2823D-08 -1.8395D-10

 G = -6.3328D+02 -1.5087D+03 -2.2300D+03 -2.8358D+03  1.5734D+05 -4.9446D+03
     -4.0928D+03  8.1552D+02 -2.5922D+02 -1.4977D+03 -1.0016D+03 -6.2022D+02
      3.5820D+03  1.5733D+03 -1.3010D+04  1.8799D+02


ITERATION     4

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    8.8365558097578073E-008

At iterate    4    f=  2.31139D+04    |proj g|=  5.32118D+05

 X =  4.1944D-09  9.9893D-09  1.4765D-08  1.8775D-08  1.2284D-07  3.2744D-08
      2.7108D-08 -5.3915D-09  1.7227D-09  9.9152D-09  6.6315D-09  4.1062D-09
     -2.3673D-08 -1.0209D-08  8.6159D-08 -1.2393D-09
```

```
 G = -6.3259D+02 -1.5081D+03 -2.2291D+03 -2.8353D+03  5.3212D+05 -4.9409D+03
     -4.0883D+03  8.1782D+02 -2.5707D+02 -1.4974D+03 -1.0012D+03 -6.2010D+02
      3.5946D+03  1.6398D+03 -1.3000D+04  1.8965D+02


ITERATION     5

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.3456306291274544E-007

At iterate    5    f=  2.31139D+04    |proj g|=  1.28780D+06

 X =  1.7712D-08  4.2183D-08  6.2351D-08  7.9283D-08  1.2501D-07  1.3827D-07
      1.1447D-07 -2.2770D-08  7.2726D-09  4.1871D-08  2.8004D-08  1.7340D-08
     -9.9983D-08 -4.3181D-08  3.6383D-07 -5.2352D-09

 G = -6.3120D+02 -1.5068D+03 -2.2274D+03 -2.8343D+03  1.2878D+06 -4.9335D+03
     -4.0793D+03  8.2246D+02 -2.5274D+02 -1.4968D+03 -1.0005D+03 -6.1984D+02
      3.6201D+03  1.7739D+03 -1.2979D+04  1.9300D+02


ITERATION     6

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    8.9083116877568545E-007

At iterate    6    f=  2.31139D+04    |proj g|=  2.41823D+06

 X =  5.3705D-08  1.2791D-07  1.8906D-07  2.4040D-07  1.2825D-07  4.1926D-07
      3.4709D-07 -6.9046D-08  2.2050D-08  1.2696D-07  8.4913D-08  5.2578D-08
     -3.0317D-07 -1.3098D-07  1.1032D-06 -1.5875D-08

 G = -6.2911D+02 -1.5050D+03 -2.2249D+03 -2.8328D+03  2.4182D+06 -4.9223D+03
     -4.0659D+03  8.2941D+02 -2.4625D+02 -1.4960D+03 -9.9937D+02 -6.1946D+02
      3.6582D+03  1.9745D+03 -1.2948D+04  1.9801D+02


ITERATION     7
```

```
----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.5279011834938847E-006

At iterate    7    f=  2.31139D+04    |proj g|=  4.30422D+06

 X =  1.5584D-07  3.7116D-07  5.4862D-07  6.9760D-07  1.3365D-07  1.2166D-06
      1.0072D-06 -2.0036D-07  6.3985D-08  3.6842D-07  2.4640D-07  1.5257D-07
     -8.7977D-07 -3.8011D-07  3.2013D-06 -4.6068D-08

 G = -6.2564D+02 -1.5019D+03 -2.2206D+03 -2.8303D+03  4.3042D+06 -4.9037D+03
     -4.0434D+03  8.4099D+02 -2.3542D+02 -1.4946D+03 -9.9753D+02 -6.1883D+02
      3.7218D+03  2.3092D+03 -1.2895D+04  2.0637D+02


ITERATION     8

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    6.7501273852657410E-006

At iterate    8    f=  2.31138D+04    |proj g|=  7.32020D+06

 X =  4.2858D-07  1.0207D-06  1.5087D-06  1.9184D-06  1.4228D-07  3.3458D-06
      2.7698D-06 -5.5100D-07  1.7596D-07  1.0132D-06  6.7762D-07  4.1958D-07
     -2.4194D-06 -1.0454D-06  8.8037D-06 -1.2669D-07

 G = -6.2008D+02 -1.4970D+03 -2.2137D+03 -2.8262D+03  7.3202D+06 -4.8739D+03
     -4.0075D+03  8.5951D+02 -2.1809D+02 -1.4924D+03 -9.9458D+02 -6.1781D+02
      3.8234D+03  2.8448D+03 -1.2811D+04  2.1975D+02


ITERATION     9

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.8075492157081601E-005

At iterate    9    f=  2.31137D+04    |proj g|=  1.22204D+07
```

```
 X =  1.1589D-06  2.7601D-06  4.0797D-06  5.1876D-06  1.5627D-07  9.0473D-06
      7.4898D-06 -1.4900D-06  4.7581D-07  2.7397D-06  1.8323D-06  1.1346D-06
     -6.5423D-06 -2.8268D-06  2.3806D-05 -3.4258D-07

 G = -6.1105D+02 -1.4889D+03 -2.2025D+03 -2.8196D+03  1.2220D+07 -4.8253D+03
     -3.9489D+03  8.8960D+02 -1.8990D+02 -1.4888D+03 -9.8979D+02 -6.1616D+02
      3.9883D+03  3.7157D+03 -1.2673D+04  2.4151D+02


ITERATION    10

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    4.7764259490236209E-005

At iterate   10    f=  2.31133D+04    |proj g|=  2.01293D+07

 X =  3.0888D-06  7.3564D-06  1.0873D-05  1.3826D-05  1.7876D-07  2.4114D-05
      1.9962D-05 -3.9711D-06  1.2682D-06  7.3020D-06  4.8837D-06  3.0240D-06
     -1.7437D-05 -7.5342D-06  6.3449D-05 -9.1308D-07

 G = -5.9648D+02 -1.4759D+03 -2.1843D+03 -2.8088D+03  2.0129D+07 -4.7466D+03
     -3.8541D+03  9.3816D+02 -1.4431D+02 -1.4830D+03 -9.8205D+02 -6.1350D+02
      4.2542D+03  5.1229D+03 -1.2446D+04  2.7666D+02


ITERATION    11

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.2586733304802177E-004

At iterate   11    f=  2.31123D+04    |proj g|=  3.29085D+07

 X =  8.1744D-06  1.9469D-05  2.8776D-05  3.6591D-05  2.1489D-07  6.3816D-05
      5.2830D-05 -1.0509D-05  3.3562D-06  1.9324D-05  1.2924D-05  8.0028D-06
     -4.6147D-05 -1.9939D-05  1.6792D-04 -2.4164D-06

 G = -5.7294D+02 -1.4548D+03 -2.1549D+03 -2.7912D+03  3.2909D+07 -4.6184D+03
     -3.7002D+03  1.0166D+03 -7.0394D+01 -1.4735D+03 -9.6951D+02 -6.0920D+02
      4.6829D+03  7.4011D+03 -1.2070D+04  3.3357D+02
```

```
ITERATION    12

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.2999506718716786E-004

At iterate   12    f=  2.31097D+04    |proj g|=  5.34724D+07

 X =  2.1508D-05  5.1224D-05  7.5713D-05  9.6274D-05  2.7247D-07  1.6791D-04
      1.3900D-04 -2.7652D-05  8.8304D-06  5.0845D-05  3.4006D-05  2.1056D-05
     -1.2142D-04 -5.2462D-05  4.4180D-04 -6.3579D-06

 G = -5.3509D+02 -1.4205D+03 -2.1070D+03 -2.7624D+03  5.3472D+07 -4.4095D+03
     -3.4502D+03  1.1429D+03  4.9192D+01 -1.4580D+03 -9.4927D+02 -6.0228D+02
      5.3703D+03  1.1079D+04 -1.1443D+04  4.2543D+02


ITERATION    13

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    8.6046149006244362E-004

At iterate   13    f=  2.31029D+04    |proj g|=  8.62983D+07

 X =  5.6274D-05  1.3403D-04  1.9810D-04  2.5190D-04  3.6294D-07  4.3932D-04
      3.6369D-04 -7.2349D-05  2.3104D-05  1.3303D-04  8.8975D-05  5.5093D-05
     -3.1768D-04 -1.3726D-04  1.1560D-03 -1.6635D-05

 G = -4.7474D+02 -1.3652D+03 -2.0296D+03 -2.7149D+03  8.6298D+07 -4.0696D+03
     -3.0457D+03  1.3444D+03  2.4178D+02 -1.4327D+03 -9.1681D+02 -5.9123D+02
      6.4615D+03  1.6980D+04 -1.0380D+04  5.7282D+02


ITERATION    14

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------
```

```
 LINE SEARCH           0  times; norm of step =    2.2172296756586439E-003

At iterate   14    f=  2.30856D+04    |proj g|=  1.37552D+08

 X =  1.4586D-04  3.4739D-04  5.1347D-04  6.5291D-04  5.0048D-07  1.1387D-03
      9.4267D-04 -1.8753D-04  5.9886D-05  3.4482D-04  2.3062D-04  1.4280D-04
     -8.2342D-04 -3.5578D-04  2.9962D-03 -4.3118D-05

 G = -3.8067D+02 -1.2771D+03 -1.9059D+03 -2.6372D+03  1.3755D+08 -3.5215D+03
     -2.3995D+03  1.6590D+03  5.4692D+02 -1.3918D+03 -8.6570D+02 -5.7398D+02
      8.1492D+03  2.6275D+04 -8.5593D+03  8.0495D+02


ITERATION    15

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    5.5548891178574082E-003

At iterate   15    f=  2.30425D+04    |proj g|=  2.13061D+08

 X =  3.7030D-04  8.8193D-04  1.3036D-03  1.6576D-03  6.9361D-07  2.8909D-03
      2.3932D-03 -4.7608D-04  1.5204D-04  8.7540D-04  5.8548D-04  3.6253D-04
     -2.0905D-03 -9.0325D-04  7.6066D-03 -1.0947D-04

 G = -2.4249D+02 -1.1428D+03 -1.7164D+03 -2.5131D+03  2.1306D+08 -2.6692D+03
     -1.4094D+03  2.1226D+03  1.0080D+03 -1.3277D+03 -7.8932D+02 -5.4857D+02
      1.0594D+04  4.0178D+04 -5.4617D+03  1.1521D+03


ITERATION    16

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.2977148019832284E-002

At iterate   16    f=  2.29435D+04    |proj g|=  3.07751D+08

 X =  8.9464D-04  2.1307D-03  3.1494D-03  4.0046D-03  9.1160D-07  6.9842D-03
      5.7819D-03 -1.1502D-03  3.6731D-04  2.1149D-03  1.4145D-03  8.7586D-04
     -5.0505D-03 -2.1822D-03  1.8377D-02 -2.6446D-04
```

```
 G = -7.0193D+01 -9.6278D+02 -1.4601D+03 -2.3332D+03  3.0775D+08 -1.4857D+03
     -7.0916D+01  2.7033D+03  1.6160D+03 -1.2378D+03 -6.9088D+02 -5.1675D+02
      1.3548D+04  5.8126D+04 -5.1618D+02  1.6004D+03


ITERATION    17

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.5747134264788408E-002

At iterate   17    f=  2.27540D+04    |proj g|=  3.77268D+08

 X =  1.9349D-03  4.6083D-03  6.8116D-03  8.6613D-03  1.0086D-06  1.5106D-02
      1.2505D-02 -2.4877D-03  7.9443D-04  4.5742D-03  3.0593D-03  1.8943D-03
     -1.0923D-02 -4.7197D-03  3.9747D-02 -5.7199D-04

 G =  5.4008D+01 -7.9975D+02 -1.2244D+03 -2.1371D+03  3.7727D+08 -3.2089D+02
      1.1578D+03  3.1247D+03  2.1385D+03 -1.1470D+03 -6.1230D+02 -4.9353D+02
      1.5401D+04  7.2507D+04  5.8391D+03  1.9612D+03


ITERATION    18

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.6147542417118275E-002

At iterate   18    f=  2.25071D+04    |proj g|=  3.26551D+08

 X =  3.3955D-03  8.0868D-03  1.1953D-02  1.5199D-02  7.2474D-07  2.6508D-02
      2.1944D-02 -4.3654D-03  1.3941D-03  8.0269D-03  5.3685D-03  3.3242D-03
     -1.9168D-02 -8.2822D-03  6.9748D-02 -1.0037D-03

 G = -4.3478D+01 -8.1430D+02 -1.2385D+03 -2.0664D+03  3.2655D+08 -1.8489D+02
      1.0748D+03  2.7949D+03  2.0098D+03 -1.1306D+03 -6.4926D+02 -5.1080D+02
      1.2963D+04  6.5840D+04  1.0256D+04  1.8006D+03


ITERATION    19
```

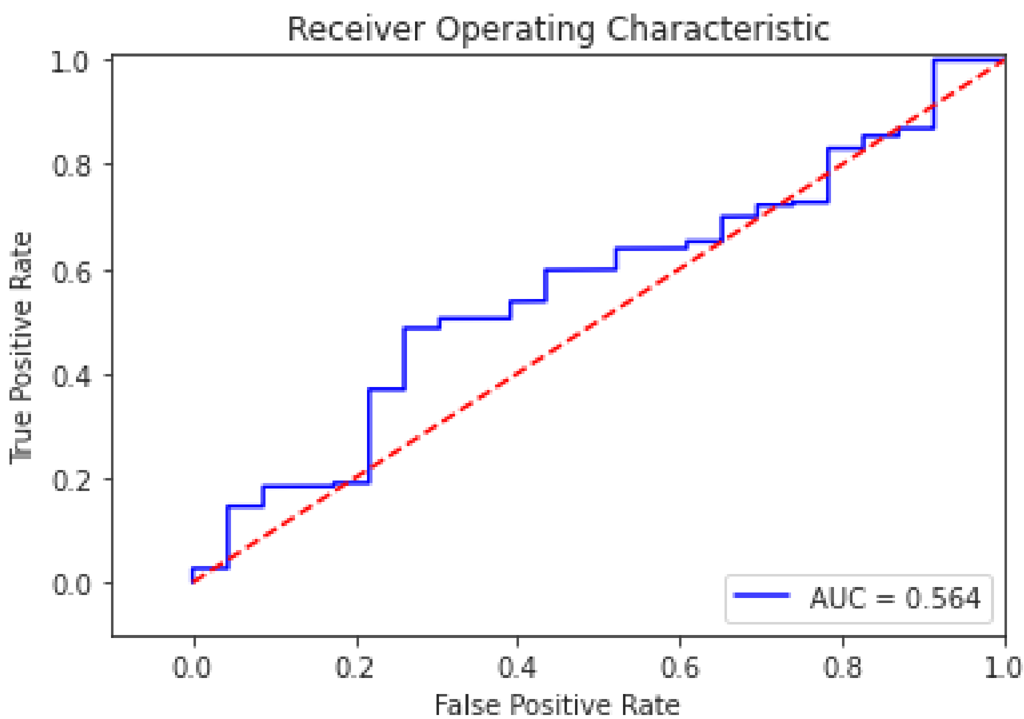
```
----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    2.4716490820984583E-002

At iterate   19    f=  2.23553D+04    |proj g|=  1.53027D+08

 X =  4.3941D-03  1.0465D-02  1.5469D-02  1.9669D-02  1.2338D-07  3.4304D-02
      2.8398D-02 -5.6493D-03  1.8041D-03  1.0388D-02  6.9475D-03  4.3019D-03
     -2.4806D-02 -1.0718D-02  9.0262D-02 -1.2989D-03

 G = -3.6504D+02 -1.0515D+03 -1.5662D+03 -2.2057D+03  1.5303D+08 -1.4700D+03
     -6.4689D+02  1.7140D+03  1.1203D+03 -1.2214D+03 -8.1086D+02 -5.6898D+02
      6.6274D+03  3.6517D+04  9.3447D+03  1.0721D+03


ITERATION    20

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.8930027746198238E-003

At iterate   20    f=  2.23232D+04    |proj g|=  3.55989D+07

 X =  4.4706D-03  1.0647D-02  1.5738D-02  2.0012D-02 -2.3554D-07  3.4901D-02
      2.8893D-02 -5.7477D-03  1.8355D-03  1.0569D-02  7.0684D-03  4.3768D-03
     -2.5238D-02 -1.0905D-02  9.1834D-02 -1.3216D-03

 G = -5.8210D+02 -1.2397D+03 -1.8262D+03 -2.3539D+03  3.5599D+07 -2.5838D+03
     -2.0073D+03  9.9363D+02  4.5545D+02 -1.3033D+03 -9.2439D+02 -6.0827D+02
      2.6435D+03  1.5945D+04  6.5571D+03  5.5652D+02


ITERATION    21

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    3.9279214218366186E-003

At iterate   21    f=  2.23195D+04    |proj g|=  9.44705D+05
```

```
 X =  4.3119D-03  1.0269D-02  1.5179D-02  1.9301D-02 -3.2546D-07  3.3662D-02
      2.7867D-02 -5.5436D-03  1.7703D-03  1.0193D-02  6.8175D-03  4.2214D-03
     -2.4342D-02 -1.0518D-02  8.8573D-02 -1.2746D-03

 G = -6.4612D+02 -1.3034D+03 -1.9144D+03 -2.4136D+03  9.4471D+05 -2.9851D+03
     -2.4670D+03  7.8390D+02  2.4060D+02 -1.3336D+03 -9.5908D+02 -6.1988D+02
      1.5526D+03  9.6463D+03  5.0849D+03  3.9754D+02


ITERATION    22

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.4958486473720335E-003

At iterate   22    f=  2.23193D+04    |proj g|=  7.81593D+05

 X =  4.2515D-03  1.0125D-02  1.4966D-02  1.9031D-02 -3.2517D-07  3.3190D-02
      2.7476D-02 -5.4659D-03  1.7455D-03  1.0051D-02  6.7220D-03  4.1622D-03
     -2.4001D-02 -1.0370D-02  8.7332D-02 -1.2568D-03

 G = -6.4929D+02 -1.3089D+03 -1.9222D+03 -2.4212D+03 -7.8159D+05 -3.0262D+03
     -2.5068D+03  7.7423D+02  2.2452D+02 -1.3369D+03 -9.6114D+02 -6.2047D+02
      1.5227D+03  9.2651D+03  4.8222D+03  3.8762D+02


ITERATION    23

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

 LINE SEARCH           0  times; norm of step =    1.3371981880801531E-004

At iterate   23    f=  2.23193D+04    |proj g|=  9.39512D+03

 X =  4.2461D-03  1.0113D-02  1.4947D-02  1.9006D-02 -3.2236D-07  3.3148D-02
      2.7442D-02 -5.4590D-03  1.7433D-03  1.0038D-02  6.7134D-03  4.1569D-03
     -2.3970D-02 -1.0357D-02  8.7221D-02 -1.2552D-03

 G = -6.4785D+02 -1.3079D+03 -1.9207D+03 -2.4206D+03 -4.0032D+03 -3.0207D+03
     -2.4993D+03  7.7907D+02  2.2842D+02 -1.3365D+03 -9.6042D+02 -6.2020D+02
      1.5514D+03  9.3951D+03  4.8232D+03  3.9085D+02
```

```
ITERATION    24

----------------SUBSM entered-----------------


----------------exit SUBSM --------------------

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    6.5s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    6.5s finished
 LINE SEARCH           0  times; norm of step =    2.5831880621849152E-007

At iterate   24    f=  2.23193D+04    |proj g|=  9.39630D+03

 X =  4.2461D-03  1.0113D-02  1.4947D-02  1.9006D-02 -3.2234D-07  3.3148D-02
      2.7441D-02 -5.4590D-03  1.7433D-03  1.0038D-02  6.7134D-03  4.1569D-03
     -2.3970D-02 -1.0357D-02  8.7221D-02 -1.2552D-03

 G = -6.4784D+02 -1.3079D+03 -1.9207D+03 -2.4206D+03  2.7460D+03 -3.0207D+03
     -2.4992D+03  7.7912D+02  2.2846D+02 -1.3365D+03 -9.6042D+02 -6.2020D+02
      1.5516D+03  9.3963D+03  4.8234D+03  3.9088D+02

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
   16     24     32      1     0     0   9.396D+03   2.232D+04

 X =  4.2461D-03  1.0113D-02  1.4947D-02  1.9006D-02 -3.2234D-07  3.3148D-02
      2.7441D-02 -5.4590D-03  1.7433D-03  1.0038D-02  6.7134D-03  4.1569D-03
     -2.3970D-02 -1.0357D-02  8.7221D-02 -1.2552D-03
  F =   22319.297138709000

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
```

[96]:

```
y_pred_sample_score = tmp.decision_function(X_test)


fpr, tpr, thresholds = roc_curve(y_test, y_pred_sample_score)

roc_auc = auc(fpr,tpr)

# Plot ROC
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b',label='AUC = %0.3f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.0])
plt.ylim([-0.1,1.01])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



18 Even with SMOTE oversampling the model is not performing well

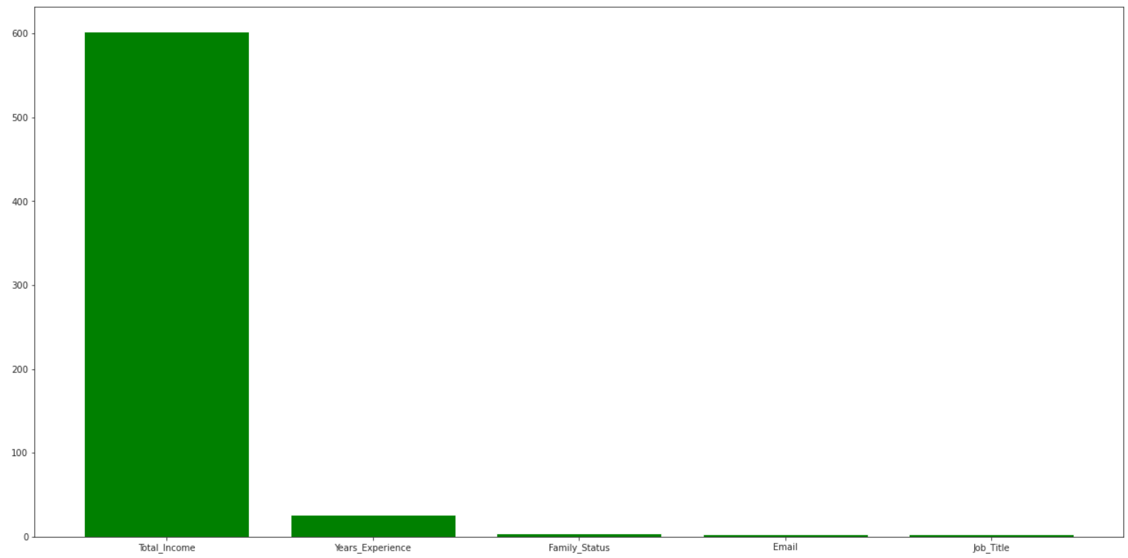
19 7- Best features with SelectKBest

[97]:
```
# Lets see if the selected features are the best
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# find best scored 5 features
selector = SelectKBest(chi2, k=5).fit(X_train, y_train)
```

[98]:
```
print('Score list:', selector.scores_)
print('Feature list:', X_train.columns)
```

```
Score list: [1.77334941e+00 2.15334793e-01 3.38194460e-01 2.34851253e+00
 6.01593589e+02 1.82976536e+00 1.34056848e+00 4.33055719e+00
 5.12877141e-01 9.49605496e-02 4.83232373e-01 2.57547465e+00
 2.56887071e+00 1.76578254e+00 2.60111565e+01]
Feature list: Index(['Gender', 'Car_Owner', 'Realty_Owner', 'Children',
'Total_Income',
       'Income_Type', 'Education_Type', 'Family_Status', 'Housing_Type',
       'Work_Phone', 'Phone', 'Email', 'Job_Title', 'Age', 'Years_Experience'],
      dtype='object')
```

[99]:
```
indices = np.argsort(selector.scores_)[::-1]

# To get your top 5 feature names
features = []
for i in range(5):
    features.append(X_train.columns[indices[i]])

# Now plot
plt.figure()
plt.bar(features, selector.scores_[indices[range(5)]], color='g')
dev_check=plt.gcf()
dev_check.set_size_inches(22,11)
plt.show()
```



[100]:
```
cols = selector.get_support(indices=True)
features_df_new = X_train.iloc[:,cols]
```

[101]:
```
features_df_new
```

[101]:

	Total_Income	Family_Status	Email	Job_Title	Years_Experience
8929	63000.00	1	0	6	5.00
9621	180000.00	1	0	8	11.00
8891	157500.00	1	1	3	2.00
4585	202500.00	0	0	8	6.00
17194	270000.00	1	0	8	4.00
…	…	…	…	…	…
21575	166500.00	1	0	11	16.00
5390	157500.00	0	0	4	4.00
860	180000.00	1	0	14	1.00
15795	90000.00	1	0	10	1.00
23654	112500.00	2	0	8	6.00

[16734 rows x 5 columns]

20 8- Kmeans clustering

[102]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

[103]:
```
df.isnull().sum()
```

[103]:
```
ID                    0
Gender                0
Car_Owner             0
Realty_Owner          0
Children              0
Total_Income          0
Income_Type           0
Education_Type        0
Family_Status         0
Housing_Type          0
Work_Phone            0
Phone                 0
Email                 0
Job_Title             0
Age                   0
Years_Experience      0
Status                0
dtype: int64
```

[104]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23906 entries, 0 to 23905
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                23906 non-null  int64
 1   Gender            23906 non-null  int64
 2   Car_Owner         23906 non-null  int64
 3   Realty_Owner      23906 non-null  int64
 4   Children          23906 non-null  int64
 5   Total_Income      23906 non-null  float64
 6   Income_Type       23906 non-null  int8
 7   Education_Type    23906 non-null  int8
 8   Family_Status     23906 non-null  int8
 9   Housing_Type      23906 non-null  int8
 10  Work_Phone        23906 non-null  int64
 11  Phone             23906 non-null  int64
 12  Email             23906 non-null  int64
 13  Job_Title         23906 non-null  int8
 14  Age               23906 non-null  float64
 15  Years_Experience  23906 non-null  float64
 16  Status            23906 non-null  int64
dtypes: float64(3), int64(9), int8(5)
memory usage: 2.5 MB
```

[105]:
```
X = df.drop(['Status','ID'], axis=1, inplace=False)

y = df['Status']
```

21 9- Feature Scaling

[106]:
```
cols = X.columns
```

[107]:
```
from sklearn.preprocessing import MinMaxScaler

mns = MinMaxScaler()

X = mns.fit_transform(X)
```

[108]:
```
X = pd.DataFrame(X, columns=[cols])
```

[109]:
```
X.head()
```

[109]:
```
  Gender Car_Owner Realty_Owner Children Total_Income Income_Type  \
0   1.00      1.00         1.00     0.00         0.17        1.00
1   0.00      0.00         1.00     0.00         0.50        0.00
2   0.00      0.00         1.00     0.00         0.50        0.00
3   0.00      0.00         1.00     0.00         0.50        0.00
4   0.00      0.00         1.00     0.00         0.50        0.00

  Education_Type Family_Status Housing_Type Work_Phone Phone Email Job_Title  \
0           1.00          0.25         0.20       0.00  0.00  0.00      0.94
1           1.00          0.75         0.20       0.00  1.00  1.00      0.82
2           1.00          0.75         0.20       0.00  1.00  1.00      0.82
3           1.00          0.75         0.20       0.00  1.00  1.00      0.82
4           1.00          0.75         0.20       0.00  1.00  1.00      0.82

   Age Years_Experience
0 0.81             0.12
1 0.68             0.32
2 0.68             0.32
3 0.68             0.32
4 0.68             0.32
```

22 Kmeans

[110]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=42)
```

```
kmeans.fit(X)
```

[110]: KMeans(n_clusters=2, random_state=42)

[111]:
```
kmeans.cluster_centers_
```

[111]:
```
array([[5.78253048e-01, 1.00000000e+00, 6.56406156e-01, 2.61343194e-01,
        3.70638860e-01, 6.71072357e-01, 7.38606836e-01, 3.02443534e-01,
        2.67559464e-01, 2.64041575e-01, 2.79132520e-01, 1.01239256e-01,
        4.37919601e-01, 4.05186675e-01, 2.36630022e-01],
       [2.41870504e-01, 5.00155473e-14, 6.49208633e-01, 2.19748201e-01,
        2.98374948e-01, 6.70593525e-01, 7.86007194e-01, 3.48273381e-01,
        2.65179856e-01, 2.83093525e-01, 3.02158273e-01, 1.00791367e-01,
        4.65010580e-01, 4.30142354e-01, 2.54854676e-01]])
```

[112]:
```
kmeans.inertia_
```

[112]: 39070.99126786699

[113]:
```
labels = kmeans.labels_

# check how many of the samples were correctly labeled
correct_labels = sum(y == labels)

print("Result: %d out of %d samples were correctly labeled." % (correct_labels,␣
 ↪y.size))
```

Result: 13888 out of 23906 samples were correctly labeled.

[114]:
```
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

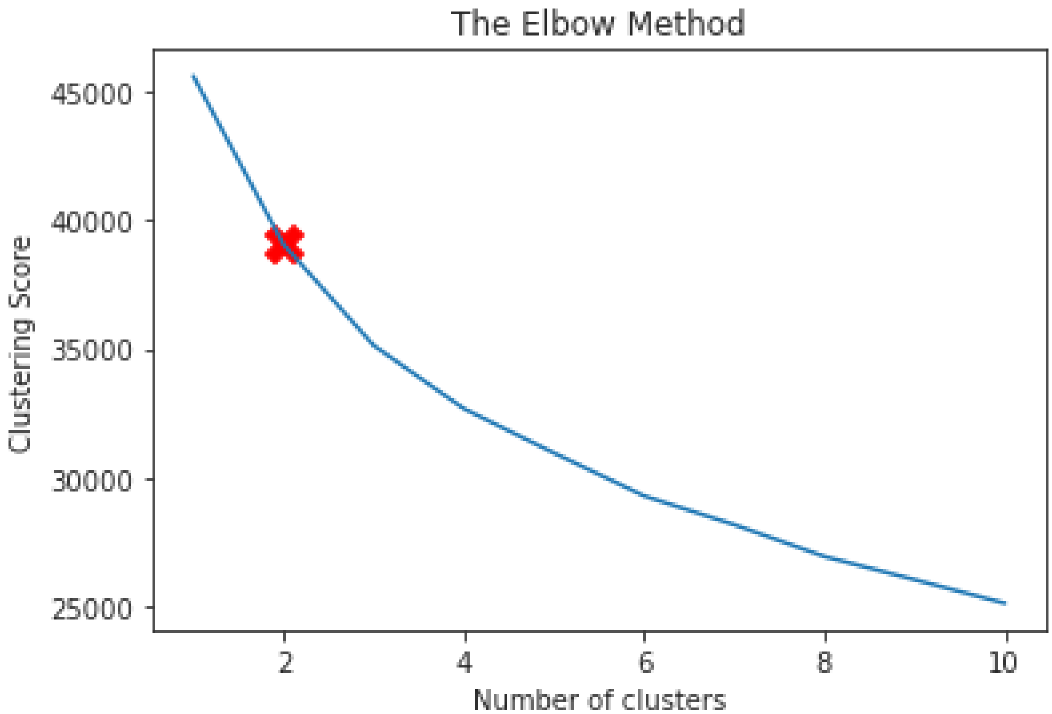
Accuracy score: 0.58

[115]:
```
f1_score(y_test, y_predict, labels=None, pos_label=1, average='binary',␣
 ↪sample_weight=None, zero_division='warn')
```

[115]: 0.9973430289470004

[116]:
```
#Elbow method to measure how our model works
cs = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init␣
 ↪= 10, random_state = 0)
    kmeans.fit(X)
    cs.append(kmeans.inertia_) # inertia_ = Sum of squared distances of samples␣
 ↪to their closest cluster center.
plt.plot(range(1, 11), cs)
```

```
plt.scatter(2,cs[1], s = 200, c = 'red', marker='X')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Clustering Score')
plt.show()
```



22.1 Compare the Silhouette scores for clusters up to 10

[117]:
```
#silhouette score needs to be near 1 rather than -1
from sklearn.metrics import silhouette_score, silhouette_samples

for n_clusters in range(2,11):
    km = KMeans (n_clusters=n_clusters)
    preds = km.fit_predict(df)
    centers = km.cluster_centers_

    score = silhouette_score(df, preds, metric='euclidean')
    print ("For n_clusters = {}, silhouette score is {}".format(n_clusters,␣
 ↪score))
```

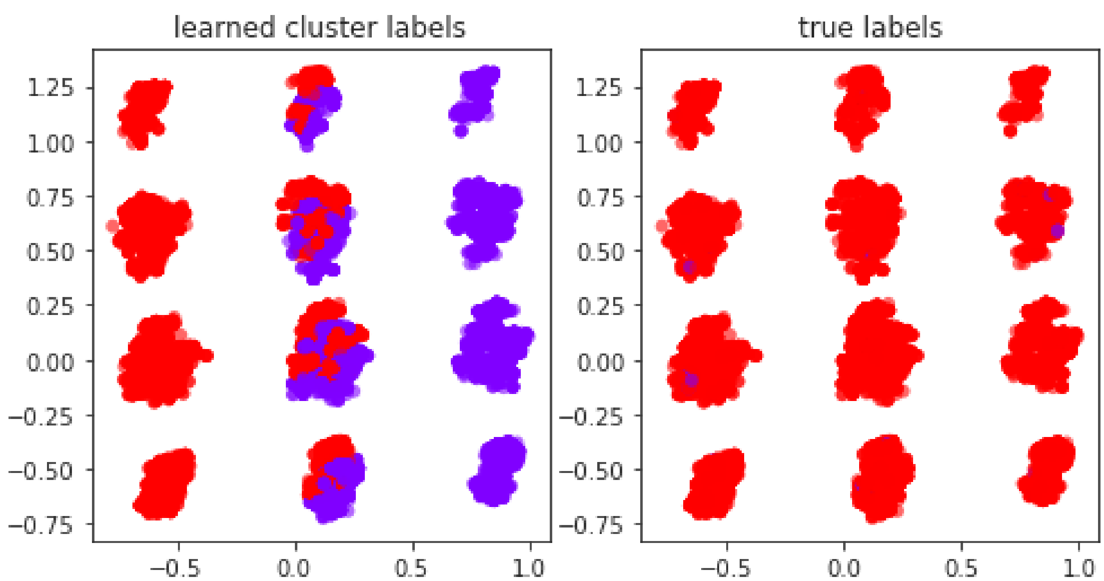
```
For n_clusters = 2, silhouette score is 0.4655578254479565
For n_clusters = 3, silhouette score is 0.37983935312155875
For n_clusters = 4, silhouette score is 0.385698979883062
```

```
For n_clusters = 5, silhouette score is 0.39590169268464354
[CV 2/5] END …C=1.0;, score=0.591 total time=   0.2s
[CV 1/5] END …C=2.0;, score=0.500 total time=   0.1s
[CV 3/5] END …C=2.0;, score=0.500 total time=   0.0s
[CV 5/5] END …C=2.0;, score=0.597 total time=   0.2s
[CV 2/5] END …C=3.0;, score=0.591 total time=   0.2s
[CV 1/5] END …C=4.0;, score=0.500 total time=   0.1s
[CV 4/5] END …C=4.0;, score=0.594 total time=   0.3s
[CV 4/5] END …C=5.0;, score=0.594 total time=   0.2s
[CV 3/5] END …C=6.0;, score=0.500 total time=   0.0s
[CV 5/5] END …C=6.0;, score=0.597 total time=   0.1s
[CV 2/5] END …C=7.0;, score=0.591 total time=   0.3s
[CV 1/5] END …C=8.0;, score=0.500 total time=   0.1s
[CV 3/5] END …C=8.0;, score=0.500 total time=   0.1s
[CV 5/5] END …C=8.0;, score=0.597 total time=   0.2s
[CV 2/5] END …C=9.0;, score=0.591 total time=   0.3s
[CV 1/5] END …C=10.0;, score=0.500 total time=   0.0s
[CV 2/5] END …C=10.0;, score=0.591 total time=   0.2s
[CV 3/5] END …C=1.0;, score=0.500 total time=   0.1s
[CV 4/5] END …C=1.0;, score=0.594 total time=   0.2s
[CV 4/5] END …C=2.0;, score=0.594 total time=   0.2s
[CV 3/5] END …C=3.0;, score=0.500 total time=   0.1s
[CV 5/5] END …C=3.0;, score=0.597 total time=   0.2s
[CV 2/5] END …C=4.0;, score=0.591 total time=   0.2s
[CV 1/5] END …C=5.0;, score=0.500 total time=   0.0s
[CV 3/5] END …C=5.0;, score=0.500 total time=   0.1s
[CV 5/5] END …C=5.0;, score=0.597 total time=   0.1s
[CV 1/5] END …C=6.0;, score=0.500 total time=   0.1s
[CV 4/5] END …C=6.0;, score=0.594 total time=   0.2s
[CV 3/5] END …C=7.0;, score=0.500 total time=   0.1s
[CV 5/5] END …C=7.0;, score=0.597 total time=   0.2s
[CV 2/5] END …C=8.0;, score=0.591 total time=   0.3s
[CV 1/5] END …C=9.0;, score=0.500 total time=   0.1s
[CV 3/5] END …C=9.0;, score=0.500 total time=   0.0s
[CV 4/5] END …C=9.0;, score=0.594 total time=   0.3s
[CV 4/5] END …C=10.0;, score=0.594 total time=   0.2s
[CV 1/5] END …C=1.0;, score=0.500 total time=   0.1s
[CV 5/5] END …C=1.0;, score=0.597 total time=   0.1s
[CV 2/5] END …C=2.0;, score=0.591 total time=   0.3s
[CV 1/5] END …C=3.0;, score=0.500 total time=   0.0s
[CV 4/5] END …C=3.0;, score=0.594 total time=   0.2s
[CV 3/5] END …C=4.0;, score=0.500 total time=   0.1s
[CV 5/5] END …C=4.0;, score=0.597 total time=   0.1s
[CV 2/5] END …C=5.0;, score=0.591 total time=   0.2s
[CV 2/5] END …C=6.0;, score=0.591 total time=   0.2s
[CV 1/5] END …C=7.0;, score=0.500 total time=   0.1s
[CV 4/5] END …C=7.0;, score=0.594 total time=   0.3s
[CV 4/5] END …C=8.0;, score=0.594 total time=   0.4s
```

```
[CV 5/5] END …C=9.0;, score=0.597 total time=   0.2s
[CV 3/5] END …C=10.0;, score=0.500 total time=   0.0s
[CV 5/5] END …C=10.0;, score=0.597 total time=   0.2s
For n_clusters = 6, silhouette score is 0.3826523647127884
For n_clusters = 7, silhouette score is 0.39012701517667386
For n_clusters = 8, silhouette score is 0.37478061008684715
For n_clusters = 9, silhouette score is 0.38188459144681486
For n_clusters = 10, silhouette score is 0.3666494679647772
```

23 10- PCA Decomposition

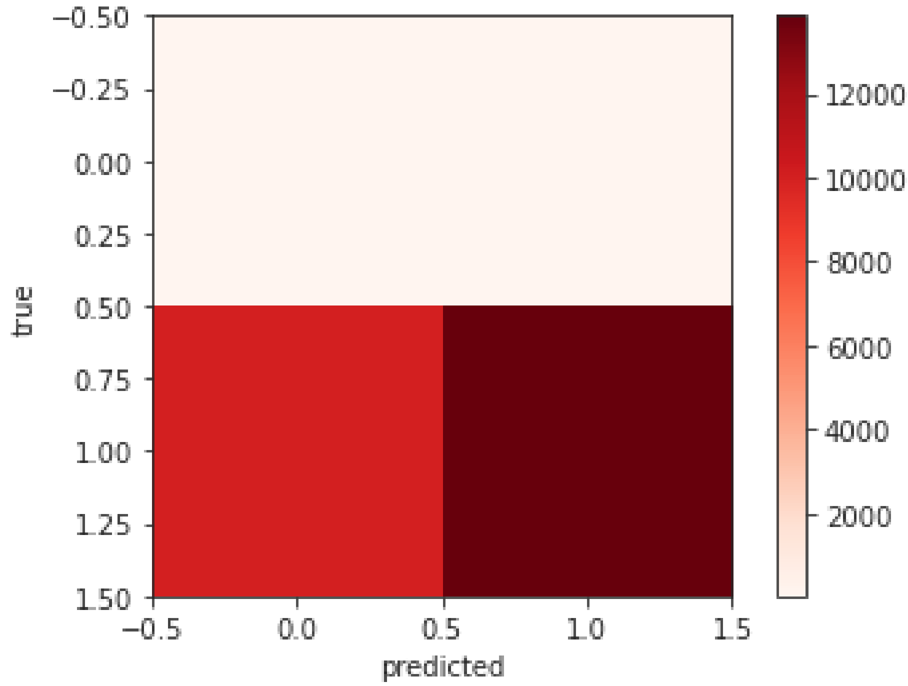
[118]:
```
from sklearn.decomposition import PCA

X = PCA(2).fit_transform(X)

kwargs = dict(cmap = plt.cm.get_cmap('rainbow', 2),
              edgecolor='none', alpha=0.6)
fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].scatter(X[:, 0], X[:, 1], c=labels, **kwargs)
ax[0].set_title('learned cluster labels')

ax[1].scatter(X[:, 0], X[:, 1], c=y, **kwargs)
ax[1].set_title('true labels');
```



[119]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y, labels))
```

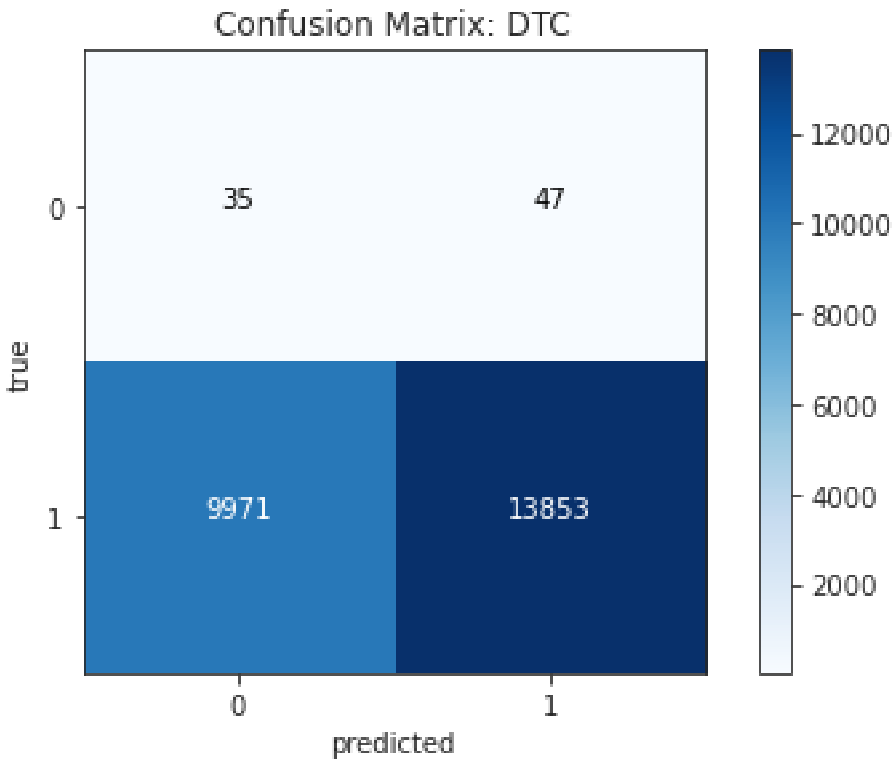
```
plt.imshow(confusion_matrix(y, labels),
           cmap='Reds', interpolation='nearest')
plt.colorbar()
plt.grid(False)
plt.ylabel('true')
plt.xlabel('predicted');
```

```
[[   35    47]
 [ 9971 13853]]
```



[120]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y, labels))
vis_conf_mat(confusion_matrix(y, labels), classes=class_names, normalize =
 ↪False,
                      title='Confusion Matrix: DTC')

plt.grid(False)
plt.ylabel('true')
plt.xlabel('predicted');
```

```
[[   35    47]
 [ 9971 13853]]
[[   35    47]
 [ 9971 13853]]
```



[121]:
```
print(classification_report(y, labels))
```

```
              precision    recall  f1-score   support

           0       0.00      0.43      0.01        82
           1       1.00      0.58      0.73     23824

    accuracy                           0.58     23906
   macro avg       0.50      0.50      0.37     23906
weighted avg       0.99      0.58      0.73     23906
```

24 EXTRA Gaussian Mixture Model

[122]:
```
## Import the GaussianMixture class
from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=2, random_state=123, n_init=10)
preds = gm.fit_predict(X)
correct_labels = sum(y == preds)
print("Result: %d out of %d samples were correctly labeled." % (correct_labels,␣
 ↪y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 13362 out of 23906 samples were correctly labeled.
Accuracy score: 0.56
```

[ ]:

```

```