# Introduction

Creditworthiness is the parameter that decides whether a person or company will be considered to be worthy or deserving to be given financial credit for certain period of time based on their previous repayment history.

Financial institutions uses credit score for evaluating and quantifying to decide that an applicant is worthy to be given credit.

The worth obtained using creditworthiness is used to decide the interest rates on credit and credit limit (the amount to be sanctioned) for the existing borrower.

# Objective

The objective here is to build a model to

1. To take credit decisions based on individual characteristics
2. To give an early warning to potential credit defauls

## Importing important libraries

Here we are going to import the import libraries used for my project. The basic libraries are Pandas, Numpy, Sklearn, Mathplotlib etc. Initially, we have a notion of deciding whether a borrower is worthy or not. So, we can say that here we are going to address classification problem. So, I also imported Logistic regression model from Scikit learn. Logistic regression is a model which is used when our output class is binary (discrete). Logistic regression is used to model the probability of each class.

Also, we need to evaluate the performance of our model, so I have imported metric from Scikit learn.

In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
# Importing library for logistic regression
from sklearn.linear_model import LogisticRegression

# Importing performance metrics - accuracy score & confusion matrix
from sklearn.metrics import accuracy_score,confusion_matrix, f1_score
```

The Seaborn is a Python library for data visualization based on matplotlib. So, it is good to use it here.

In [2]:

```python
import seaborn as sns
```

## Getting familiar with data

Importing the data and doing preliminiary analysis

In [3]:

```python
df = pd.read_excel('CreditWorthiness.xlsx',sheet_name='Data')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Cbal         1000 non-null   object
 1   Cdur         1000 non-null   int64 
 2   Chist        1000 non-null   object
 3   Cpur         1000 non-null   object
 4   Camt         1000 non-null   int64 
 5   Sbal         1000 non-null   object
 6   Edur         1000 non-null   object
 7   InRate       1000 non-null   int64 
 8   MSG          1000 non-null   object
 9   Oparties     1000 non-null   object
 10  Rdur         1000 non-null   object
 11  Prop         1000 non-null   object
 12  age          1000 non-null   int64 
 13  inPlans      1000 non-null   object
 14  Htype        1000 non-null   object
 15  NumCred      1000 non-null   int64 
 16  JobType      1000 non-null   object
 17  Ndepend      1000 non-null   int64 
 18  telephone    1000 non-null   object
 19  foreign      1000 non-null   object
 20  creditScore  1000 non-null   object
dtypes: int64(6), object(15)
memory usage: 164.2+ KB
```

It looks like the data has no NAN values. Therefore, there is no need for NAN value removal and data imputation. The target variable seems to be credit score.

The data has six numerical and 15 categorical variables. Looking top ten entries to see whether the data has any special characters

The variable description is given below

| Variables | Data Type | Description |
|---|---|---|
| Cbal | CATEGORICAL | Balance in the checking account in Rs. (rupees) but as part of categories |
| Cdur | NUMERICAL | The duration of the credit in months (numerical) |
| Chist | CATEGORICAL | The pattern of credit dues payment over time, whether the borrower has paid his monthly dues promptly , has fallen behind currently or in the past, or in a critical state. |
| Cpur | CATEGORICAL | Purpose of the loan – for a variety of needs from vehicles, furniture to housing |
| Camt | NUMERICAL | The actual amount under credit/loan in Rs. (rupees) (numerical) |
| Sbal | CATEGORICAL | Balance in the savings bank account in Rs. (rupees) but as part of categories |
| Edur | CATEGORICAL | Duration of employment, need to understand from borrower's background if he is unemployed or employed, if employed then for how many years. |
| InRate | NUMERICAL | The instalment rate provided as percentage of disposable income (we do not know what the disposable income could mean here and is best to take the rate as is.) (numerical) |
| MSG | CATEGORICAL | MSG (marital status and gender) informs us on the person's gender and current status. |
| Oparties | CATEGORICAL | Other parties i.e. the presence of a guarantor or co-applicant to the loan/credit |
| Rdur | CATEGORICAL | Duration in current residence in years but as part of categories |
| Prop | CATEGORICAL | Other properties that the person possess. Note : Person could possess a car/similar property in addition to the ones under Cpur |
| Age | NUMERICAL | Age of the borrower |
| InPlans | CATEGORICAL | Other instalment plans that the borrower may have i.e from another bank or stores |
| Htype | CATEGORICAL | Type of housing, whether the borrower owns the property, pays rent or lives for free. |
| NumCred | NUMERICAL | Number of existing credits at the bank (numerical) |
| JobType | CATEGORICAL | Type of job the borrower is employed in whether skilled or unskilled, management or self employed etc. |
| Ndepend | NUMERICAL | Number of dependents (numerical) |
| Tel | CATEGORICAL | If the borrower has a telephone number or not ? |
| Foreign | CATEGORICAL | If the borrower is foreign worker or not ? |
| **CreditScore** | **CATEGORICAL** | **Chances of borrower closing his credit promptly, Categorised as 'Good' and 'Bad'** |

The glimpse of data is given below containing five top rows

In [4]:

```
df.head()
```

Out[4]:

| | Cbal | Cdur | Chist | Cpur | Camt | Sbal | Edur | InRate | MSG | Oparties | ... | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 <= Rs. < 2000 | 9 | all settled till now | Business | 13790 | Rs. < 1000 | 1 to 4 years | 2 | married or widowed male | no one | ... | |
| **1** | 0 <= Rs. < 2000 | 15 | dues not paid earlier | electronics | 15250 | no savings account | more than 7 years | 4 | single male | yes, guarantor | ... | |
| **2** | 0 <= Rs. < 2000 | 36 | none taken/all settled | Business | 19410 | Rs. < 1000 | more than 7 years | 4 | single male | no one | ... | |
| **3** | 0 <= Rs. < 2000 | 48 | none taken/all settled | Business | 144090 | Rs. < 1000 | 1 to 4 years | 2 | single male | no one | ... | |
| **4** | no checking account | 24 | all settled till now | electronics | 31690 | Rs. < 1000 | less than 1 year | 4 | divorced or separated or married female | no one | ... | insu |

5 rows × 21 columns

The glimse of data containing the last five rows of dataset

In [5]:
```python
df.tail()
```

Out[5]:

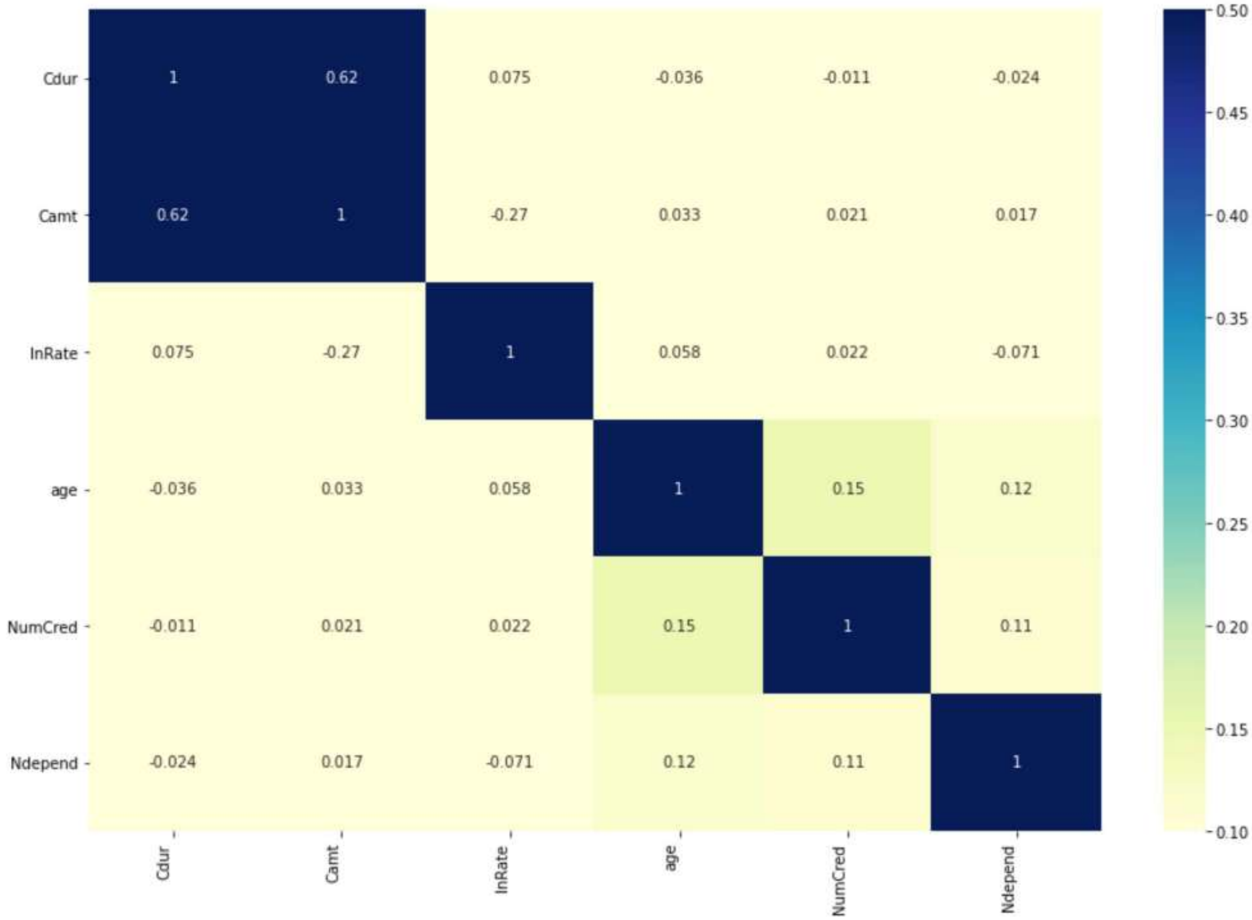
| | Cbal | Cdur | Chist | Cpur | Camt | Sbal | Edur | InRate | MSG | Oparties | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **995** | no checking account | 6 | all settled | second hand vehicle | 7710 | no savings account | 1 to 4 years | 1 | single male | yes, guarantor | ... |
| **996** | 0 <= Rs. < 2000 | 12 | all settled till now | electronics | 64560 | no savings account | not employed | 2 | single male | no one | ... |
| **997** | no checking account | 36 | dues not paid earlier | electronics | 95540 | Rs. < 1000 | 1 to 4 years | 2 | divorced or separated or married female | no one | ... |
| **998** | Rs. >=2000 | 18 | all settled till now | second hand vehicle | 19490 | Rs. < 1000 | more than 7 years | 3 | divorced or separated or married female | no one | ... |
| **999** | Rs. < 0 | 36 | dues not paid earlier | furniture | 62170 | Rs. < 1000 | less than 1 year | 4 | divorced or separated or married female | yes, co-applicant | ... |

5 rows × 21 columns

The heatmap given below will give the relation of Credit score with numerical variables

In [6]:
```python
plt.figure(figsize=(15,10))
#plot heat map
g=sns.heatmap(df.corr(),annot=True,vmin=0.1,vmax=0.5,cmap="YlGnBu")
g.set_xticklabels(g.get_xticklabels(), rotation=90, horizontalalignment='right')
g.set_yticklabels(g.get_yticklabels(), rotation=0, horizontalalignment='right')
```

Out[6]:
```
[Text(0, 0.5, 'Cdur'),
 Text(0, 1.5, 'Camt'),
 Text(0, 2.5, 'InRate'),
 Text(0, 3.5, 'age'),
 Text(0, 4.5, 'NumCred'),
 Text(0, 5.5, 'Ndepend')]
```




The five number summary of numerical variables of dataset is given below

In [7]:

```
df.describe()
```

Out[7]:

| | Cdur | Camt | InRate | age | NumCred | Ndepend |
|---|---|---|---|---|---|---|
| **count** | 1000.000000 | 1000.00000 | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 20.903000 | 32592.58000 | 2.973000 | 35.546000 | 1.407000 | 1.155000 |
| **std** | 12.058814 | 28227.36876 | 1.118715 | 11.375469 | 0.577654 | 0.362086 |
| **min** | 4.000000 | 2380.00000 | 1.000000 | 19.000000 | 1.000000 | 1.000000 |
| **25%** | 12.000000 | 13535.00000 | 2.000000 | 27.000000 | 1.000000 | 1.000000 |
| **50%** | 18.000000 | 23075.00000 | 3.000000 | 33.000000 | 1.000000 | 1.000000 |
| **75%** | 24.000000 | 39602.50000 | 4.000000 | 42.000000 | 2.000000 | 1.000000 |
| **max** | 72.000000 | 184120.00000 | 4.000000 | 75.000000 | 4.000000 | 2.000000 |

The statistical summary of categorical variables of dataset is given below

In [8]:

```
df.describe(include='object')
```

Out[8]:

| | Cbal | Chist | Cpur | Sbal | Edur | MSG | Oparties | Rdur | Prop | inPlans | Htype | Jol |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | 1000 | |

| | Cbal | Chist | Cpur | Sbal | Edur | MSG | Oparties | Rdur | Prop | inPlans | Htype | Jol |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **unique** | 4 | 4 | 10 | 5 | 5 | 4 | 3 | 4 | 4 | 3 | 3 | |
| **top** | no checking account | all settled till now | electronics | Rs. < 1000 | 1 to 4 years | single male | no one | more than 3 years | Other cars etc. | none | own | emp c pc |
| **freq** | 394 | 618 | 280 | 603 | 339 | 548 | 907 | 413 | 332 | 814 | 713 | |

Let us see how many bad or good credit score are there

In [9]:

```
df.groupby('creditScore').size()
```

Out[9]:

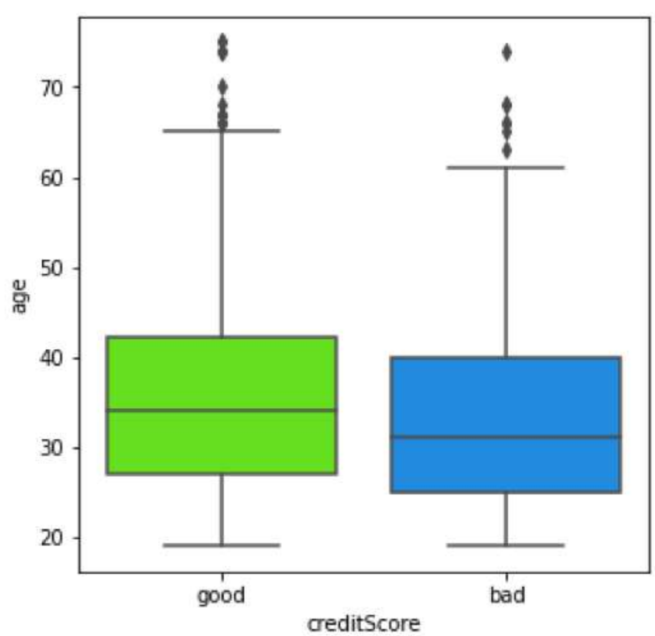
```
creditScore
bad      300
good     700
dtype: int64
```

There are 300 people with bad credit score and 700 people with good credit score. Hard encoding bad as zero and good as one.

In [10]:

```
plt.figure(figsize=(5,5))
sns.boxplot(df["creditScore"],df["age"],palette="gist_rainbow")
```

Out[10]: <AxesSubplot:xlabel='creditScore', ylabel='age'>



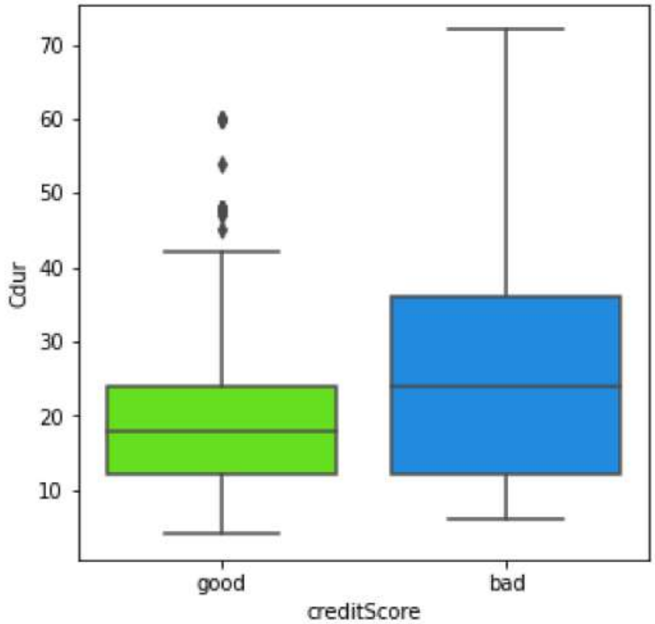

This graphs shows that older people have good credit score

If the installment rate is lower than

In [11]:

```
plt.figure(figsize=(5,5))
sns.boxplot(df["creditScore"],df["Cdur"],palette="gist_rainbow")
```

Out[11]: <AxesSubplot:xlabel='creditScore', ylabel='Cdur'>

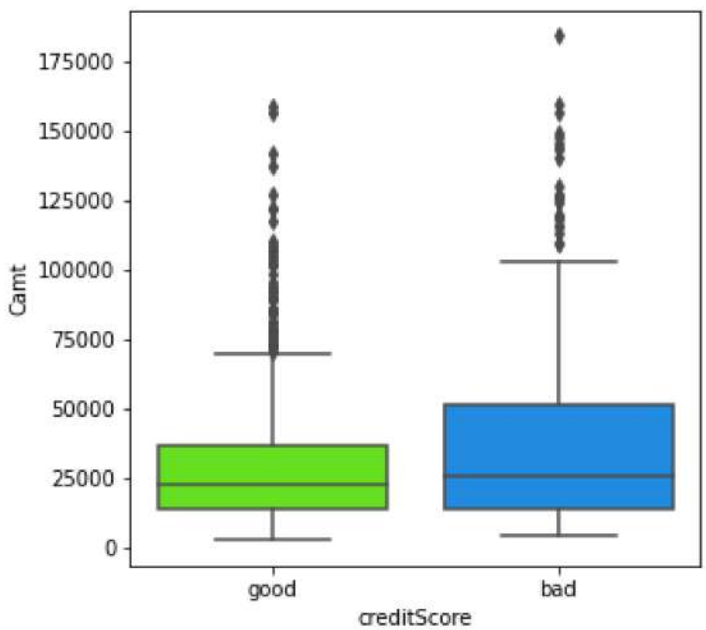



The borrowers with higher duration of credit has bad credit score

In [12]:

```
plt.figure(figsize=(5,5))
sns.boxplot(df["creditScore"],df["Camt"],palette="gist_rainbow")
```

Out[12]: <AxesSubplot:xlabel='creditScore', ylabel='Camt'>




The borrower with larger amount of credit have bad credit score

In [13]:

```
df['creditScore']=df['creditScore'].map({'bad':0,
                                          'good':1})
```

Encoding categorical variables into dummy variables

In [14]:
```
X = pd.get_dummies(df)
```

In [15]:
```
df_columns_list=list(X.columns)
```

In [16]:
```
# Separating the input names from species
features=list(set(df_columns_list)-set(['creditScore']))
```

## Separating dependent and independent varriables

In [17]:
```
# Storing the output values in y
target=list(['creditScore'])
y=X[target].values
```

In [18]:
```
Xf=X[features]
Xf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 63 columns):
 #   Column                                                  Non-Null Count
Dtype
---  ------                                                  --------------
-----
 0   JobType_non resident either unemployed or  unskilled    1000 non-null
uint8
 1   Edur_more than 7 years                                  1000 non-null
uint8
 2   Cpur_new vehicle                                        1000 non-null
uint8
 3   Cpur_education                                          1000 non-null
uint8
 4   Edur_1 to 4 years                                       1000 non-null
uint8
 5   foreign_no                                              1000 non-null
uint8
 6   Camt                                                    1000 non-null
int64
 7   MSG_divorced or separated male                          1000 non-null
uint8
 8   Rdur_more than 3 years                                  1000 non-null
uint8
 9   Edur_not employed                                       1000 non-null
uint8
 10  MSG_single male                                         1000 non-null
uint8
 11  Sbal_no savings account                                 1000 non-null
uint8
 12  Edur_less than 1 year                                   1000 non-null
uint8
 13  Chist_dues not paid earlier                             1000 non-null
uint8
 14  Cbal_0 <= Rs. < 2000                                    1000 non-null
uint8
 15  NumCred                                                 1000 non-null
int64
 16  Rdur_less than a year                                   1000 non-null
```

```
uint8
 17  Cbal_Rs. >=2000                                                   1000 non-null
uint8
 18  MSG_divorced or separated or married female                       1000 non-null
uint8
 19  Prop_life insurance/building society                              1000 non-null
uint8
 20  Cbal_no checking account                                          1000 non-null
uint8
 21  Ndepend                                                           1000 non-null
int64
 22  Oparties_no one                                                   1000 non-null
uint8
 23  Sbal_Rs. >= 10,000                                                1000 non-null
uint8
 24  Cbal_ Rs. < 0                                                     1000 non-null
uint8
 25  Prop_real estate                                                  1000 non-null
uint8
 26  telephone_yes                                                     1000 non-null
uint8
 27  Sbal_1000 <= Rs. < 5,000                                          1000 non-null
uint8
 28  inPlans_stores                                                    1000 non-null
uint8
 29  Cpur_electronics                                                  1000 non-null
uint8
 30  Oparties_yes, co-applicant                                        1000 non-null
uint8
 31  Rdur_1 to 2 years                                                 1000 non-null
uint8
 32  JobType_resident unskilled                                        1000 non-null
uint8
 33  Chist_all settled till now                                        1000 non-null
uint8
 34  Edur_4 to 7 years                                                 1000 non-null
uint8
 35  telephone_no                                                      1000 non-null
uint8
 36  Chist_none taken/all settled                                      1000 non-null
uint8
 37  InRate                                                            1000 non-null
int64
 38  Cpur_miscellaneous                                                1000 non-null
uint8
 39  JobType_employed either in management, self or in high position  1000 non-null
uint8
 40  Htype_pays rent                                                   1000 non-null
uint8
 41  Chist_all settled                                                 1000 non-null
uint8
 42  Sbal_Rs. < 1000                                                   1000 non-null
uint8
 43  JobType_employee with official position                           1000 non-null
uint8
 44  Oparties_yes, guarantor                                           1000 non-null
uint8
 45  Htype_free                                                        1000 non-null
uint8
 46  Htype_own                                                         1000 non-null
uint8
 47  Cpur_renovation                                                   1000 non-null
uint8
```

```
 48  Rdur_2 to 3 years                                    1000 non-null
uint8
 49  Prop_Other cars etc.                                 1000 non-null
uint8
 50  age                                                  1000 non-null
int64
 51  Cpur_retaining                                       1000 non-null
uint8
 52  inPlans_bank                                         1000 non-null
uint8
 53  Cpur_second hand vehicle                             1000 non-null
uint8
 54  foreign_yes                                          1000 non-null
uint8
 55  Cpur_domestic needs                                  1000 non-null
uint8
 56  MSG_married or widowed male                          1000 non-null
uint8
 57  Cpur_Business                                        1000 non-null
uint8
 58  inPlans_none                                         1000 non-null
uint8
 59  Cpur_furniture                                       1000 non-null
uint8
 60  Sbal_5000 <= Rs. < 10,000                            1000 non-null
uint8
 61  Prop_Unknown                                         1000 non-null
uint8
 62  Cdur                                                 1000 non-null
int64
dtypes: int64(6), uint8(57)
memory usage: 102.7 KB
```

In [19]:

```
x = X[features].values
```

## The data is split into train and test data and standardization to have zero mean and unit variance

In [20]:

```
# Splitting the data into train and test
train_x, test_x, train_y, test_y = train_test_split( x, y, test_size=0.25, random_sta


# Data scaling
scaler = StandardScaler()

# Fit on training set only.
scaler.fit(train_x)
```

Out[20]:

```
StandardScaler()
```

In [21]:

```
# Apply transform to both the training set and the test set.
train_x = scaler.transform(train_x)

test_x = scaler.transform(test_x)
```

## I prefer logistic regression model to classify the output as good or

## bad

The logistic regression model is built and output are predicted using the built model

In [22]:
```
# Make an instance of the Model
logistic = LogisticRegression(penalty='l1', tol=0.01, solver='saga')

# Fitting the values for x and y
logistic.fit(train_x/np.std(train_x,0),train_y)
print(logistic.coef_)
```

```
[[ 0.00000000e+00  1.30039637e-02  2.62331889e-01 -8.82253914e-02
  -1.95099249e-02 -1.07937717e-01 -3.22763290e-01 -1.34845953e-01
   2.30386590e-05 -2.33714201e-02  1.96550839e-01  2.09384826e-01
  -2.96939085e-02  3.02857106e-01 -1.41841627e-01 -1.63549593e-01
   1.73795142e-01  3.32284064e-02 -8.67336150e-02 -1.63116260e-05
   4.24515844e-01 -3.18359856e-02 -3.62646784e-02  1.97378308e-01
  -3.15356142e-01  6.47309752e-02  6.14311669e-02 -4.45044469e-02
   2.80078758e-02  1.10654510e-01 -9.34781907e-02 -8.84666032e-02
   1.24719874e-04 -6.82694595e-02  1.50246027e-01 -6.14311669e-02
  -1.29169901e-01 -3.82734974e-01  1.03471760e-01  1.16206480e-02
  -8.00388054e-02 -2.01798800e-01 -2.74989180e-01 -7.21542749e-02
   2.30309687e-01  9.19732886e-04  2.74667446e-02 -6.59903109e-02
  -6.12380017e-03 -2.12328064e-05  2.42151241e-01  8.27436480e-04
  -1.79396760e-01 -2.86935378e-01  1.07937717e-01  7.93450720e-04
  -5.34318174e-06 -6.72335265e-02  1.00015997e-01 -7.36623329e-06
   6.69317151e-02 -1.66458551e-01 -2.59019380e-01]]
```

In [23]:
```
# Prediction from test data
prediction = logistic.predict(test_x)
```

## The evaluation metric used are confusion matrix

The accuracy is obtained and shown

In [24]:
```
# Confusion matrix
confusion_matrix = confusion_matrix(prediction,test_y)
print(confusion_matrix)

# Calculating the accuracy
accuracy_score=accuracy_score(prediction,test_y)
print(accuracy_score)
```

```
[[ 40  20]
 [ 37 153]]
0.772
```

It become really tough to say that the model is very good as the accuracy is 77.2%

In [25]:
```
# Calculating the f1_score
f1_score=f1_score(prediction,test_y)
print(f1_score)
```

```
0.8429752066115701
```

But, if we look the $F_1$ score then we can say that our basic model is really good

In [26]:

```
# Printing the misclassified values from prediction
print('Misclassified samples: %d' % (test_y != prediction).sum())
```

```
Misclassified samples: 25010
```

# Conclusion

In the above study we have used the logistic regression model for classifying the bad or good credit score. Though, we can explore more models for classification but I am limiting myself because Logistic regression is the simplest model for classification and we are getting good results here.

We have classified credit score as good or bad for the dataset available