# Project 1

# Apply

logistic Regression SVM Decision Tree RandomForest on the Loan dataset and check were you will get the best possible accuracy project note : Dependent Variable is Loan Status

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
Loan_df = pd.read_csv("C:/Users/Lenovo/Documents/Data Set/loan.csv")
```

In [4]:

```
Loan_df
```

Out[4]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coa |
|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **609** | LP002978 | Female | No | 0 | Graduate | No | 2900 | |
| **610** | LP002979 | Male | Yes | 3+ | Graduate | No | 4106 | |
| **611** | LP002983 | Male | Yes | 1 | Graduate | No | 8072 | |
| **612** | LP002984 | Male | Yes | 2 | Graduate | No | 7583 | |
| **613** | LP002990 | Female | No | 0 | Graduate | Yes | 4583 | |

614 rows × 13 columns

In [5]:

```
Loan_df.head()
```

Out[5]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coapp |
|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0 | Graduate | No | 5849 | |
| **1** | LP001003 | Male | Yes | 1 | Graduate | No | 4583 | |
| **2** | LP001005 | Male | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | LP001006 | Male | Yes | 0 | Not Graduate | No | 2583 | |
| **4** | LP001008 | Male | No | 0 | Graduate | No | 6000 | |

In [6]:

```
Loan_df.tail()
```

Out[6]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Coa |
|---|---|---|---|---|---|---|---|---|
| **609** | LP002978 | Female | No | 0 | Graduate | No | 2900 | |
| **610** | LP002979 | Male | Yes | 3+ | Graduate | No | 4106 | |
| **611** | LP002983 | Male | Yes | 1 | Graduate | No | 8072 | |
| **612** | LP002984 | Male | Yes | 2 | Graduate | No | 7583 | |
| **613** | LP002990 | Female | No | 0 | Graduate | Yes | 4583 | |

In [7]:

```
Loan_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [8]:

```
Loan_df.isnull().sum()
```

Out[8]:

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [9]:

```
Loan_df = Loan_df.drop("Loan_ID",axis = 1)
```

In [10]:

```
Loan_df
```

Out[10]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | Male | No | 0 | Graduate | No | 5849 | |
| **1** | Male | Yes | 1 | Graduate | No | 4583 | 15 |
| **2** | Male | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | Male | Yes | 0 | Not Graduate | No | 2583 | 23 |
| **4** | Male | No | 0 | Graduate | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | Female | No | 0 | Graduate | No | 2900 | |
| **610** | Male | Yes | 3+ | Graduate | No | 4106 | |
| **611** | Male | Yes | 1 | Graduate | No | 8072 | 2 |
| **612** | Male | Yes | 2 | Graduate | No | 7583 | |
| **613** | Female | No | 0 | Graduate | Yes | 4583 | |

614 rows × 12 columns

In [11]:

```
set(Loan_df["Education"])
```

Out[11]:

```
{'Graduate', 'Not Graduate'}
```

In [12]:

```
set(Loan_df["Married"])
```

Out[12]:

```
{'No', 'Yes', nan}
```

In [13]:

```
set(Loan_df["Gender"])
```

Out[13]:

```
{'Female', 'Male', nan}
```

In [14]:

```
set(Loan_df["Dependents"])
```

Out[14]:

```
{'0', '1', '2', '3+', nan}
```

In [15]:

```
set(Loan_df["Self_Employed"])
```

Out[15]:

```
{'No', 'Yes', nan}
```

In [16]:

```
set(Loan_df["Property_Area"])
```

Out[16]:

```
{'Rural', 'Semiurban', 'Urban'}
```

In [17]:

```
set(Loan_df["Loan_Status"])
```

Out[17]:

```
{'N', 'Y'}
```

In [18]:

```
Loan_df
```

Out[18]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | Male | No | 0 | Graduate | No | 5849 | |
| **1** | Male | Yes | 1 | Graduate | No | 4583 | 15 |
| **2** | Male | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | Male | Yes | 0 | Not Graduate | No | 2583 | 23 |
| **4** | Male | No | 0 | Graduate | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | Female | No | 0 | Graduate | No | 2900 | |
| **610** | Male | Yes | 3+ | Graduate | No | 4106 | |
| **611** | Male | Yes | 1 | Graduate | No | 8072 | 2 |
| **612** | Male | Yes | 2 | Graduate | No | 7583 | |
| **613** | Female | No | 0 | Graduate | Yes | 4583 | |

614 rows × 12 columns

In [19]:

```
df = Loan_df[['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',"ApplicantInc
df[0:5]
```

Out[19]:

```
array([['Male', 'No', '0', 'Graduate', 'No', 5849, 0.0, nan, 360.0, 1.0,
        'Urban', 'Y'],
       ['Male', 'Yes', '1', 'Graduate', 'No', 4583, 1508.0, 128.0, 360.0,
        1.0, 'Rural', 'N'],
       ['Male', 'Yes', '0', 'Graduate', 'Yes', 3000, 0.0, 66.0, 360.0,
        1.0, 'Urban', 'Y'],
       ['Male', 'Yes', '0', 'Not Graduate', 'No', 2583, 2358.0, 120.0,
        360.0, 1.0, 'Urban', 'Y'],
       ['Male', 'No', '0', 'Graduate', 'No', 6000, 0.0, 141.0, 360.0,
        1.0, 'Urban', 'Y']], dtype=object)
```

# Apply Label Encoding

In [20]:

```
from sklearn.preprocessing import LabelEncoder
```

In [21]:

```
le_Gender = LabelEncoder()
print(le_Gender)
Loan_df["Gender"] = le_Gender.fit_transform(Loan_df["Gender"])
Loan_df
```

```
LabelEncoder()
```

Out[21]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | No | 0 | Graduate | No | 5849 | |
| **1** | 1 | Yes | 1 | Graduate | No | 4583 | 15 |
| **2** | 1 | Yes | 0 | Graduate | Yes | 3000 | |
| **3** | 1 | Yes | 0 | Not Graduate | No | 2583 | 23 |
| **4** | 1 | No | 0 | Graduate | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | No | 0 | Graduate | No | 2900 | |
| **610** | 1 | Yes | 3+ | Graduate | No | 4106 | |
| **611** | 1 | Yes | 1 | Graduate | No | 8072 | 2 |
| **612** | 1 | Yes | 2 | Graduate | No | 7583 | |
| **613** | 0 | No | 0 | Graduate | Yes | 4583 | |

614 rows × 12 columns

In [22]:

```
le_Married = LabelEncoder()
print(le_Married)
Loan_df["Married"] = le_Married.fit_transform(Loan_df["Married"])
Loan_df
```

```
LabelEncoder()
```

Out[22]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | Graduate | No | 5849 | |
| **1** | 1 | 1 | 1 | Graduate | No | 4583 | 15 |
| **2** | 1 | 1 | 0 | Graduate | Yes | 3000 | |
| **3** | 1 | 1 | 0 | Not Graduate | No | 2583 | 23 |
| **4** | 1 | 0 | 0 | Graduate | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | Graduate | No | 2900 | |
| **610** | 1 | 1 | 3+ | Graduate | No | 4106 | |
| **611** | 1 | 1 | 1 | Graduate | No | 8072 | 2 |
| **612** | 1 | 1 | 2 | Graduate | No | 7583 | |
| **613** | 0 | 0 | 0 | Graduate | Yes | 4583 | |

614 rows × 12 columns

In [23]:

```
le_Education = LabelEncoder()
print(le_Education)
Loan_df["Education"] = le_Education.fit_transform(Loan_df["Education"])
Loan_df
```

```
LabelEncoder()
```

Out[23]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | No | 5849 | |
| **1** | 1 | 1 | 1 | 0 | No | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | Yes | 3000 | |
| **3** | 1 | 1 | 0 | 1 | No | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | No | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | No | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | No | 4106 | |
| **611** | 1 | 1 | 1 | 0 | No | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | No | 7583 | |
| **613** | 0 | 0 | 0 | 0 | Yes | 4583 | |

614 rows × 12 columns

In [24]:

```
le_Self_Employed = LabelEncoder()
print(le_Self_Employed)
Loan_df["Self_Employed"] = le_Self_Employed.fit_transform(Loan_df["Self_Employed"])
Loan_df
```

```
LabelEncoder()
```

Out[24]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [25]:

```
le_Property_Area = LabelEncoder()
print(le_Property_Area)
Loan_df["Property_Area"] = le_Property_Area.fit_transform(Loan_df["Property_Area"])
Loan_df
```

```
LabelEncoder()
```

Out[25]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [26]:

```
le_Loan_Status = LabelEncoder()
print(le_Loan_Status)
Loan_df["Loan_Status"] = le_Loan_Status.fit_transform(Loan_df["Loan_Status"])
Loan_df
```

```
LabelEncoder()
```

Out[26]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [27]:

```
Loan_df
```

Out[27]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [28]:

```
Loan_df.isnull().sum()
```

Out[28]:

```
Gender               0
Married              0
Dependents          15
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

In [29]:

```
Loan_df
```

Out[29]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [30]:

```
Loan_df[Loan_df['Gender'].isnull()]
```

Out[30]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome |
|---|---|---|---|---|---|---|---|

In [31]:

```
Loan_df[Loan_df['Married'].isnull()]
```

Out[31]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome |
|---|---|---|---|---|---|---|---|

In [32]:

```
Loan_df[Loan_df['Dependents'].isnull()]
```

Out[32]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **102** | 1 | 1 | NaN | 0 | 0 | 13650 | |
| **104** | 1 | 2 | NaN | 0 | 0 | 3816 | 7 |
| **120** | 1 | 1 | NaN | 0 | 0 | 5667 | 26 |
| **226** | 1 | 1 | NaN | 1 | 1 | 4735 | |
| **228** | 1 | 2 | NaN | 0 | 0 | 4758 | |
| **293** | 0 | 0 | NaN | 0 | 0 | 5417 | |
| **301** | 1 | 1 | NaN | 1 | 0 | 2875 | 17 |
| **332** | 1 | 0 | NaN | 0 | 0 | 2833 | |
| **335** | 1 | 1 | NaN | 0 | 1 | 5503 | 44 |
| **346** | 1 | 1 | NaN | 1 | 0 | 3523 | 32 |
| **355** | 0 | 0 | NaN | 0 | 0 | 3813 | |
| **435** | 0 | 2 | NaN | 0 | 0 | 10047 | |
| **517** | 1 | 1 | NaN | 1 | 0 | 3074 | 18 |
| **571** | 1 | 1 | NaN | 0 | 0 | 5116 | 14 |
| **597** | 1 | 0 | NaN | 0 | 0 | 2987 | |

In [33]:

```
Loan_df[Loan_df['Self_Employed'].isnull()]
```

Out[33]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome |
|---|---|---|---|---|---|---|---|

In [34]:

```
Loan_df[Loan_df['LoanAmount'].isnull()]
```

Out[34]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **35** | 1 | 1 | 0 | 0 | 0 | 2275 | 20 |
| **63** | 1 | 1 | 1 | 0 | 0 | 4945 | |
| **81** | 1 | 1 | 1 | 0 | 1 | 2395 | |
| **95** | 1 | 0 | 0 | 0 | 2 | 6782 | |
| **102** | 1 | 1 | NaN | 0 | 0 | 13650 | |
| **103** | 1 | 1 | 0 | 0 | 0 | 4652 | 35 |
| **113** | 0 | 0 | 1 | 0 | 1 | 7451 | |
| **127** | 1 | 0 | 0 | 0 | 0 | 3865 | 16 |
| **202** | 1 | 1 | 3+ | 1 | 0 | 3992 | |
| **284** | 1 | 1 | 0 | 0 | 0 | 20667 | |
| **305** | 1 | 0 | 0 | 1 | 0 | 2000 | |
| **322** | 1 | 1 | 2 | 1 | 0 | 3601 | 15 |
| **338** | 0 | 0 | 3+ | 1 | 0 | 1830 | |
| **387** | 1 | 1 | 0 | 1 | 0 | 3010 | 31 |
| **435** | 0 | 2 | NaN | 0 | 0 | 10047 | |
| **437** | 1 | 1 | 0 | 0 | 0 | 2213 | 11 |
| **479** | 1 | 1 | 2 | 0 | 0 | 2947 | 16 |
| **524** | 1 | 0 | 0 | 0 | 0 | 4680 | 20 |
| **550** | 1 | 1 | 2 | 0 | 1 | 6633 | |
| **551** | 1 | 1 | 1 | 1 | 0 | 2492 | 23 |
| **605** | 1 | 1 | 0 | 1 | 0 | 2400 | 38 |

In [35]:

```
Loan_df[Loan_df['Loan_Amount_Term'].isnull()]
```

Out[35]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **19** | 1 | 1 | 0 | 0 | 2 | 2600 | 35 |
| **36** | 1 | 1 | 0 | 0 | 0 | 1828 | 13 |
| **44** | 1 | 1 | 0 | 1 | 1 | 4695 | |
| **45** | 0 | 0 | 0 | 0 | 0 | 3410 | |
| **73** | 1 | 1 | 3+ | 1 | 0 | 4755 | |
| **112** | 1 | 1 | 0 | 1 | 0 | 3572 | 41 |
| **165** | 1 | 1 | 0 | 0 | 0 | 3707 | 31 |
| **197** | 0 | 0 | 0 | 1 | 0 | 1907 | 23 |
| **223** | 1 | 1 | 0 | 0 | 0 | 7578 | 10 |
| **232** | 1 | 0 | 0 | 1 | 0 | 3189 | 25 |
| **335** | 1 | 1 | NaN | 0 | 1 | 5503 | 44 |
| **367** | 1 | 0 | 0 | 0 | 0 | 5124 | |
| **421** | 0 | 0 | 0 | 1 | 0 | 2720 | |
| **423** | 1 | 1 | 1 | 0 | 0 | 7250 | 16 |

In [36]:

```
Loan_df[Loan_df['Credit_History'].isnull()]
```

Out[36]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **16** | 1 | 0 | 1 | 1 | 0 | 3596 | |
| **24** | 1 | 1 | 1 | 0 | 2 | 3717 | 29 |
| **30** | 1 | 1 | 1 | 0 | 2 | 4166 | 33 |
| **42** | 1 | 1 | 0 | 0 | 0 | 2400 | |
| **79** | 1 | 1 | 3+ | 1 | 1 | 3333 | 21 |
| **83** | 1 | 1 | 0 | 0 | 0 | 6000 | 22 |
| **86** | 1 | 1 | 2 | 1 | 0 | 3333 | 20 |
| **95** | 1 | 0 | 0 | 0 | 2 | 6782 | |
| **117** | 1 | 1 | 1 | 0 | 0 | 2214 | 13 |
| **125** | 0 | 0 | 0 | 0 | 0 | 3692 | |
| **129** | 1 | 1 | 0 | 0 | 0 | 6080 | 25 |
| **130** | 1 | 0 | 0 | 0 | 1 | 20166 | |
| **156** | 1 | 1 | 1 | 0 | 0 | 6000 | |
| **181** | 1 | 0 | 0 | 0 | 0 | 1916 | 50 |
| **187** | 1 | 1 | 0 | 0 | 0 | 2383 | 21 |
| **198** | 0 | 1 | 0 | 0 | 0 | 3416 | 28 |
| **219** | 0 | 1 | 2 | 0 | 0 | 4283 | 23 |
| **236** | 1 | 1 | 0 | 0 | 2 | 5746 | |
| **237** | 0 | 0 | 0 | 0 | 1 | 3463 | |
| **259** | 1 | 1 | 3+ | 1 | 0 | 4931 | |
| **260** | 1 | 1 | 1 | 0 | 0 | 6083 | 42 |
| **279** | 0 | 1 | 0 | 1 | 0 | 4100 | |
| **309** | 1 | 1 | 2 | 1 | 0 | 7667 | |
| **313** | 1 | 1 | 2 | 0 | 1 | 5746 | |
| **317** | 1 | 1 | 0 | 0 | 0 | 2058 | 21 |
| **318** | 0 | 0 | 1 | 0 | 0 | 3541 | |
| **323** | 0 | 0 | 0 | 0 | 0 | 3166 | 29 |
| **348** | 1 | 1 | 0 | 0 | 0 | 6333 | 45 |
| **363** | 1 | 1 | 0 | 0 | 0 | 3013 | 30 |
| **377** | 1 | 1 | 0 | 0 | 0 | 4310 | |
| **392** | 1 | 1 | 0 | 0 | 0 | 2583 | 21 |
| **395** | 1 | 1 | 2 | 0 | 0 | 3276 | 4 |
| **411** | 1 | 1 | 0 | 0 | 2 | 6256 | |
| **444** | 1 | 1 | 0 | 0 | 0 | 7333 | 83 |
| **449** | 1 | 0 | 1 | 1 | 1 | 2769 | 15 |

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **451** | 1 | 1 | 2 | 1 | 0 | 1958 | 14 |
| **460** | 2 | 1 | 0 | 0 | 1 | 2083 | 40 |
| **473** | 0 | 0 | 0 | 0 | 1 | 2500 | |
| **490** | 1 | 0 | 0 | 1 | 0 | 2699 | 27 |
| **491** | 1 | 1 | 1 | 1 | 0 | 5333 | 11 |
| **497** | 1 | 1 | 0 | 0 | 0 | 4625 | 28 |
| **503** | 1 | 1 | 1 | 1 | 0 | 4050 | 53 |
| **506** | 1 | 1 | 0 | 0 | 0 | 20833 | 66 |
| **530** | 1 | 1 | 0 | 0 | 0 | 1025 | 55 |
| **533** | 1 | 0 | 1 | 0 | 0 | 11250 | |
| **544** | 0 | 1 | 0 | 1 | 0 | 3017 | 6 |
| **556** | 0 | 0 | 0 | 0 | 0 | 2667 | 16 |
| **565** | 1 | 1 | 0 | 1 | 0 | 4467 | |
| **583** | 1 | 1 | 1 | 0 | 0 | 1880 | |
| **600** | 0 | 0 | 3+ | 0 | 2 | 416 | 416 |

In [37]:

```
Loan_df
```

Out[37]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [38]:

```
Loan_df.columns
```

Out[38]:

```
Index(['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed',
       'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

In [39]:

```
null_values_present_in_Dependents=Loan_df['Dependents'].isnull().sum()
total_no_rows = len(Loan_df)
print(f"%age of null values in Dependents column is {(null_values_present_in_Dependents/tot
```

```
%age of null values in Dependents column is 2.44299674267101%
```

In [40]:

```
null_values_present_in_LoanAmount=Loan_df['LoanAmount'].isnull().sum()
total_no_rows = len(Loan_df)
print(f"%age of null values in LoanAmount column is {(null_values_present_in_LoanAmount/tot
```

```
%age of null values in LoanAmount column is 3.5830618892508146%
```

In [41]:

```
null_values_present_in_Loan_Amount_Term=Loan_df['Loan_Amount_Term'].isnull().sum()
total_no_rows = len(Loan_df)
print(f"%age of null values in LoanAmount column is {(null_values_present_in_Loan_Amount_Te
```

```
%age of null values in LoanAmount column is 2.2801302931596092%
```

In [42]:

```
null_values_present_in_Credit_History=Loan_df['Credit_History'].isnull().sum()
total_no_rows = len(Loan_df)
print(f"%age of null values in LoanAmount column is {(null_values_present_in_Credit_History
```

```
%age of null values in LoanAmount column is 8.143322475570033%
```

# Mean/ Median /Mode imputation

In [43]:

```
Loan_df
```

Out[43]:

| | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantInc |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 5849 | |
| **1** | 1 | 1 | 1 | 0 | 0 | 4583 | 15 |
| **2** | 1 | 1 | 0 | 0 | 1 | 3000 | |
| **3** | 1 | 1 | 0 | 1 | 0 | 2583 | 23 |
| **4** | 1 | 0 | 0 | 0 | 0 | 6000 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 0 | 2900 | |
| **610** | 1 | 1 | 3+ | 0 | 0 | 4106 | |
| **611** | 1 | 1 | 1 | 0 | 0 | 8072 | 2 |
| **612** | 1 | 1 | 2 | 0 | 0 | 7583 | |
| **613** | 0 | 0 | 0 | 0 | 1 | 4583 | |

614 rows × 12 columns

In [44]:

```
## Lets go and see the percentage of missing values
Loan_df.isnull().mean()
```

Out[44]:

```
Gender               0.000000
Married              0.000000
Dependents           0.024430
Education            0.000000
Self_Employed        0.000000
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           0.035831
Loan_Amount_Term     0.022801
Credit_History       0.081433
Property_Area        0.000000
Loan_Status          0.000000
dtype: float64
```

In [45]:

```
Loan_df.dtypes
```

Out[45]:

```
Gender                int32
Married               int32
Dependents           object
Education             int32
Self_Employed         int32
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area         int32
Loan_Status           int32
dtype: object
```

In [46]:

```
# Remove the 610 row
```

In [47]:

```
Loan_df = Loan_df.drop("Dependents", axis = 1)
```

In [48]:

```
Loan_df
```

Out[48]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 11 columns

In [49]:

```
Loan_df.shape
```

Out[49]:

```
(614, 11)
```

In [50]:

```
Loan_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             614 non-null    int32  
 1   Married            614 non-null    int32  
 2   Education          614 non-null    int32  
 3   Self_Employed      614 non-null    int32  
 4   ApplicantIncome    614 non-null    int64  
 5   CoapplicantIncome  614 non-null    float64
 6   LoanAmount         592 non-null    float64
 7   Loan_Amount_Term   600 non-null    float64
 8   Credit_History     564 non-null    float64
 9   Property_Area      614 non-null    int32  
 10  Loan_Status        614 non-null    int32  
dtypes: float64(4), int32(6), int64(1)
memory usage: 38.5 KB
```

In [51]:

```
Loan_df.isnull().sum()
```

Out[51]:

```
Gender               0
Married              0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

# Mean

In [52]:

```
def impute_nan(Loan_df,variable,value):
    Loan_df[variable+"_mean"]=Loan_df[variable].fillna(value)
```

In [53]:

```
LoanAmount_mean = Loan_df.LoanAmount.mean()
```

In [54]:

```
LoanAmount_mean
```

Out[54]:

```
146.41216216216216
```

In [55]:

```
impute_nan(Loan_df,'LoanAmount',LoanAmount_mean)
```

In [56]:

```
Loan_df
```

Out[56]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 12 columns

In [57]:

```
Loan_df[Loan_df['LoanAmount'].isnull()]
```

Out[57]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **35** | 1 | 1 | 0 | 0 | 2275 | 2067.0 | |
| **63** | 1 | 1 | 0 | 0 | 4945 | 0.0 | |
| **81** | 1 | 1 | 0 | 1 | 2395 | 0.0 | |
| **95** | 1 | 0 | 0 | 2 | 6782 | 0.0 | |
| **102** | 1 | 1 | 0 | 0 | 13650 | 0.0 | |
| **103** | 1 | 1 | 0 | 0 | 4652 | 3583.0 | |
| **113** | 0 | 0 | 0 | 1 | 7451 | 0.0 | |
| **127** | 1 | 0 | 0 | 0 | 3865 | 1640.0 | |
| **202** | 1 | 1 | 1 | 0 | 3992 | 0.0 | |
| **284** | 1 | 1 | 0 | 0 | 20667 | 0.0 | |
| **305** | 1 | 0 | 1 | 0 | 2000 | 0.0 | |
| **322** | 1 | 1 | 1 | 0 | 3601 | 1590.0 | |
| **338** | 0 | 0 | 1 | 0 | 1830 | 0.0 | |
| **387** | 1 | 1 | 1 | 0 | 3010 | 3136.0 | |
| **435** | 0 | 2 | 0 | 0 | 10047 | 0.0 | |
| **437** | 1 | 1 | 0 | 0 | 2213 | 1125.0 | |
| **479** | 1 | 1 | 0 | 0 | 2947 | 1603.0 | |
| **524** | 1 | 0 | 0 | 0 | 4680 | 2087.0 | |
| **550** | 1 | 1 | 0 | 1 | 6633 | 0.0 | |
| **551** | 1 | 1 | 1 | 0 | 2492 | 2375.0 | |
| **605** | 1 | 1 | 1 | 0 | 2400 | 3800.0 | |

In [ ]:

```

```

In [58]:

```
def impute_nan(Loan_df,variable,value):
    Loan_df[variable+"_mean"]=Loan_df[variable].fillna(value)
```

In [59]:

```
Loan_Amount_Term_mean = Loan_df.Loan_Amount_Term.mean()
```

In [60]:

```
Loan_Amount_Term_mean
```

Out[60]:

342.0

In [61]:

```
impute_nan(Loan_df,'Loan_Amount_Term',Loan_Amount_Term_mean)
```

In [62]:

```
Loan_df
```

Out[62]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 13 columns

In [63]:

```
Loan_df[Loan_df['Loan_Amount_Term'].isnull()]
```

Out[63]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **19** | 1 | 1 | 0 | 2 | 2600 | 3500.0 | |
| **36** | 1 | 1 | 0 | 0 | 1828 | 1330.0 | |
| **44** | 1 | 1 | 1 | 1 | 4695 | 0.0 | |
| **45** | 0 | 0 | 0 | 0 | 3410 | 0.0 | |
| **73** | 1 | 1 | 1 | 0 | 4755 | 0.0 | |
| **112** | 1 | 1 | 1 | 0 | 3572 | 4114.0 | |
| **165** | 1 | 1 | 0 | 0 | 3707 | 3166.0 | |
| **197** | 0 | 0 | 1 | 0 | 1907 | 2365.0 | |
| **223** | 1 | 1 | 0 | 0 | 7578 | 1010.0 | |
| **232** | 1 | 0 | 1 | 0 | 3189 | 2598.0 | |
| **335** | 1 | 1 | 0 | 1 | 5503 | 4490.0 | |
| **367** | 1 | 0 | 0 | 0 | 5124 | 0.0 | |
| **421** | 0 | 0 | 1 | 0 | 2720 | 0.0 | |
| **423** | 1 | 1 | 0 | 0 | 7250 | 1667.0 | |

In [64]:

```
def impute_nan(Loan_df,variable,value):
    Loan_df[variable+"_mean"]=Loan_df[variable].fillna(value)
```

In [65]:

```
Credit_History_mean = Loan_df.Credit_History.mean()
```

In [66]:

```
Credit_History_mean
```

Out[66]:

```
0.8421985815602837
```

In [67]:

```
impute_nan(Loan_df,'Credit_History',Credit_History_mean)
```

In [68]:

```
Loan_df
```

Out[68]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 14 columns

In [69]:

```
Loan_df[Loan_df['Credit_History'].isnull()]
```

Out[69]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **16** | 1 | 0 | 1 | 0 | 3596 | 0.0 | |
| **24** | 1 | 1 | 0 | 2 | 3717 | 2925.0 | |
| **30** | 1 | 1 | 0 | 2 | 4166 | 3369.0 | |
| **42** | 1 | 1 | 0 | 0 | 2400 | 0.0 | |
| **79** | 1 | 1 | 1 | 1 | 3333 | 2166.0 | |
| **83** | 1 | 1 | 0 | 0 | 6000 | 2250.0 | |
| **86** | 1 | 1 | 1 | 0 | 3333 | 2000.0 | |
| **95** | 1 | 0 | 0 | 2 | 6782 | 0.0 | |
| **117** | 1 | 1 | 0 | 0 | 2214 | 1398.0 | |
| **125** | 0 | 0 | 0 | 0 | 3692 | 0.0 | |
| **129** | 1 | 1 | 0 | 0 | 6080 | 2569.0 | |
| **130** | 1 | 0 | 0 | 1 | 20166 | 0.0 | |
| **156** | 1 | 1 | 0 | 0 | 6000 | 0.0 | |
| **181** | 1 | 0 | 0 | 0 | 1916 | 5063.0 | |
| **187** | 1 | 1 | 0 | 0 | 2383 | 2138.0 | |
| **198** | 0 | 1 | 0 | 0 | 3416 | 2816.0 | |
| **219** | 0 | 1 | 0 | 0 | 4283 | 2383.0 | |
| **236** | 1 | 1 | 0 | 2 | 5746 | 0.0 | |
| **237** | 0 | 0 | 0 | 1 | 3463 | 0.0 | |
| **259** | 1 | 1 | 1 | 0 | 4931 | 0.0 | |
| **260** | 1 | 1 | 0 | 0 | 6083 | 4250.0 | |
| **279** | 0 | 1 | 1 | 0 | 4100 | 0.0 | |
| **309** | 1 | 1 | 1 | 0 | 7667 | 0.0 | |
| **313** | 1 | 1 | 0 | 1 | 5746 | 0.0 | |
| **317** | 1 | 1 | 0 | 0 | 2058 | 2134.0 | |
| **318** | 0 | 0 | 0 | 0 | 3541 | 0.0 | |
| **323** | 0 | 0 | 0 | 0 | 3166 | 2985.0 | |
| **348** | 1 | 1 | 0 | 0 | 6333 | 4583.0 | |
| **363** | 1 | 1 | 0 | 0 | 3013 | 3033.0 | |
| **377** | 1 | 1 | 0 | 0 | 4310 | 0.0 | |
| **392** | 1 | 1 | 0 | 0 | 2583 | 2115.0 | |
| **395** | 1 | 1 | 0 | 0 | 3276 | 484.0 | |
| **411** | 1 | 1 | 0 | 2 | 6256 | 0.0 | |
| **444** | 1 | 1 | 0 | 0 | 7333 | 8333.0 | |
| **449** | 1 | 0 | 1 | 1 | 2769 | 1542.0 | |

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **451** | 1 | 1 | 1 | 0 | 1958 | 1456.0 | |
| **460** | 2 | 1 | 0 | 1 | 2083 | 4083.0 | |
| **473** | 0 | 0 | 0 | 1 | 2500 | 0.0 | |
| **490** | 1 | 0 | 1 | 0 | 2699 | 2785.0 | |
| **491** | 1 | 1 | 1 | 0 | 5333 | 1131.0 | |
| **497** | 1 | 1 | 0 | 0 | 4625 | 2857.0 | |
| **503** | 1 | 1 | 1 | 0 | 4050 | 5302.0 | |
| **506** | 1 | 1 | 0 | 0 | 20833 | 6667.0 | |
| **530** | 1 | 1 | 0 | 0 | 1025 | 5500.0 | |
| **533** | 1 | 0 | 0 | 0 | 11250 | 0.0 | |
| **544** | 0 | 1 | 1 | 0 | 3017 | 663.0 | |
| **556** | 0 | 0 | 0 | 0 | 2667 | 1625.0 | |
| **565** | 1 | 1 | 1 | 0 | 4467 | 0.0 | |
| **583** | 1 | 1 | 0 | 0 | 1880 | 0.0 | |
| **600** | 0 | 0 | 0 | 2 | 416 | 41667.0 | |

In [70]:

```
print(Loan_df['LoanAmount_mean'].std())
```

```
84.0374676831965
```

In [71]:

```
print(Loan_df['Loan_Amount_Term_mean'].std())
```

```
64.37248862679246
```

In [72]:

```
print(Loan_df['Credit_History_mean'].std())
```

```
0.3496809866561489
```

In [73]:

```
Loan_df['LoanAmount_mean'].isnull().sum()
```

Out[73]:

```
0
```

In [74]:

```
Loan_df['Loan_Amount_Term_mean'].isnull().sum()
```

Out[74]:

```
0
```

In [75]:

```
Loan_df['Credit_History_mean'].isnull().sum()
```

Out[75]:

```
0
```

# Median

In [76]:

```
def impute_nan_median(Loan_df,variable,value):
    Loan_df[variable+"_median"]=Loan_df[variable].fillna(value)
```

In [77]:

```
median=Loan_df.LoanAmount.median()
median
print("the median from LoanAmount column which is available in df dataframe is",median)
```

```
the median from LoanAmount column which is available in df dataframe is 128.
0
```

In [78]:

```
impute_nan_median(Loan_df,'LoanAmount',median)
```

In [79]:

```
Loan_df
```

Out[79]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 15 columns

In [80]:

```
def impute_nan_median(Loan_df,variable,value):
    Loan_df[variable+"_median"]=Loan_df[variable].fillna(value)
```

In [81]:

```
median=Loan_df.Loan_Amount_Term.median()
median
print("the median from Loan_Amount_Term column which is available in df dataframe is",media
```

```
the median from Loan_Amount_Term column which is available in df dataframe i
s 360.0
```

In [82]:

```
impute_nan_median(Loan_df,'Loan_Amount_Term',median)
```

In [83]:

```
Loan_df
```

Out[83]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 16 columns

In [84]:

```
def impute_nan_median(Loan_df,variable,value):
    Loan_df[variable+"_median"]=Loan_df[variable].fillna(value)
```

In [85]:

```
median=Loan_df.Credit_History.median()
median
print("the median from Credit_History column which is available in df dataframe is",median)
```

```
the median from Credit_History column which is available in df dataframe is
1.0
```

In [86]:

```
impute_nan_median(Loan_df,'Credit_History',median)
```

In [87]:

```
Loan_df
```

Out[87]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 17 columns

In [88]:

```
print(Loan_df['LoanAmount_median'].std())
```

```
84.10723338042615
```

In [89]:

```
print(Loan_df['Loan_Amount_Term_median'].std())
```

```
64.42862906767301
```

In [90]:

```
print(Loan_df['Credit_History_median'].std())
```

```
0.3523386063583013
```

In [91]:

```
Loan_df['LoanAmount_median'].isnull().sum()
```

Out[91]:

```
0
```

In [92]:

```
Loan_df['Loan_Amount_Term_median'].isnull().sum()
```

Out[92]:

```
0
```

In [93]:

```
Loan_df['Credit_History_median'].isnull().sum()
```

Out[93]:

```
0
```

# Mode

In [94]:

```
def impute_nan_mode(Loan_df,variable,value):
    print(value)
    Loan_df[variable+"_mode"]=Loan_df[variable].fillna(value)
```

In [95]:

```
mode = Loan_df.LoanAmount.mode()
type(mode)
```

Out[95]:

```
pandas.core.series.Series
```

In [96]:

```
mode
```

Out[96]:

```
0    120.0
Name: LoanAmount, dtype: float64
```

In [97]:

```
mode[0]
```

Out[97]:

```
120.0
```

In [98]:

```
print(mode)
impute_nan_mode(Loan_df,'LoanAmount',mode[0])
Loan_df
```

```
0    120.0
Name: LoanAmount, dtype: float64
120.0
```

Out[98]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 18 columns

In [99]:

```
def impute_nan_mode(Loan_df,variable,value):
    print(value)
    Loan_df[variable+"_mode"]=Loan_df[variable].fillna(value)
```

In [100]:

```
mode = Loan_df.Loan_Amount_Term.mode()
type(mode)
```

Out[100]:

```
pandas.core.series.Series
```

In [101]:

```
mode
```

Out[101]:

```
0    360.0
Name: Loan_Amount_Term, dtype: float64
```

In [102]:

```
mode[0]
```

Out[102]:

```
360.0
```

In [103]:

```
print(mode)
impute_nan_mode(Loan_df,'Loan_Amount_Term',mode[0])
Loan_df
```

```
0    360.0
Name: Loan_Amount_Term, dtype: float64
360.0
```

Out[103]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 19 columns

In [104]:

```
def impute_nan_mode(Loan_df,variable,value):
    print(value)
    Loan_df[variable+"_mode"]=Loan_df[variable].fillna(value)
```

In [105]:

```
mode = Loan_df.Credit_History.mode()
type(mode)
```

Out[105]:

```
pandas.core.series.Series
```

In [106]:

```
mode
```

Out[106]:

```
0    1.0
Name: Credit_History, dtype: float64
```

In [107]:

```
mode[0]
```

Out[107]:

```
1.0
```

In [108]:

```
print(mode)
impute_nan_mode(Loan_df,'Credit_History',mode[0])
Loan_df
```

```
0    1.0
Name: Credit_History, dtype: float64
1.0
```

Out[108]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 20 columns

In [109]:

```
print(Loan_df['LoanAmount_mode'].std())
```

```
84.1809670102725
```

In [110]:

```
print(Loan_df['Loan_Amount_Term_mode'].std())
```

```
64.42862906767301
```

In [111]:

```
print(Loan_df['Credit_History_mode'].std())
```

```
0.3523386063583013
```

In [112]:

```
Loan_df['LoanAmount_mode'].isnull().sum()
```

Out[112]:

```
0
```

In [113]:

```
Loan_df['Loan_Amount_Term_mode'].isnull().sum()
```

Out[113]:

```
0
```

In [114]:

```
Loan_df['Credit_History_mode'].isnull().sum()
```

Out[114]:

```
0
```

# Standard Deviation

In [115]:

```
print(f"std of original {Loan_df['LoanAmount'].std()} mean {Loan_df['LoanAmount_mean'].std(
print(f"std of original {Loan_df['Loan_Amount_Term'].std()} mean {Loan_df['Loan_Amount_Term
print(f"std of original {Loan_df['Credit_History'].std()} mean {Loan_df['Credit_History_mea
```

```
std of original 85.58732523570545 mean 84.0374676831965
std of original 65.12040985461256 mean 64.37248862679246
std of original 0.3648783192364049 mean 0.3496809866561489
```

In [116]:

```
print(f"std of original {Loan_df['LoanAmount'].std()} median {Loan_df['LoanAmount_median'].
print(f"std of original {Loan_df['Loan_Amount_Term'].std()} median {Loan_df['Loan_Amount_Te
print(f"std of original {Loan_df['Credit_History'].std()} median {Loan_df['Credit_History_m
```

```
std of original 85.58732523570545 median 84.10723338042615
std of original 65.12040985461256 median 64.42862906767301
std of original 0.3648783192364049 median 0.3523386063583013
```

In [117]:

```
print(f"std of original {Loan_df['LoanAmount'].std()} mode {Loan_df['LoanAmount_mode'].std(
print(f"std of original {Loan_df['Loan_Amount_Term'].std()} mode {Loan_df['Loan_Amount_Term
print(f"std of original {Loan_df['Credit_History'].std()} mode {Loan_df['Credit_History_mod
```

```
std of original 85.58732523570545 mode 84.1809670102725
std of original 65.12040985461256 mode 64.42862906767301
std of original 0.3648783192364049 mode 0.3523386063583013
```

# Conclusion : For this example we can use the mean column because mean column having the less standard deviation as compared to the other columns.

# So,therefore we can remove the other columns and only use the mean column for building a model

In [118]:

```
Loan_df = Loan_df.drop(['LoanAmount_median','Loan_Amount_Term_median','Credit_History_media
```

In [119]:

```
Loan_df
```

Out[119]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | LoanAm |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 14 columns

In [120]:

```
Loan_df = Loan_df.drop(["LoanAmount","Loan_Amount_Term","Credit_History"],axis=1)
```

In [121]:

```
Loan_df
```

Out[121]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Property |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

614 rows × 11 columns

In [122]:

```
Loan_df.shape
```

Out[122]:

```
(614, 11)
```

# These are the columns are used for the further analysis

In [123]:

```
X = Loan_df.drop(["Loan_Status"],axis=1) #independent variable
X[0:5]
```

Out[123]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Property_A |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |

In [124]:

```
X.head()
```

Out[124]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Property_A |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 5849 | 0.0 | |
| **1** | 1 | 1 | 0 | 0 | 4583 | 1508.0 | |
| **2** | 1 | 1 | 0 | 1 | 3000 | 0.0 | |
| **3** | 1 | 1 | 1 | 0 | 2583 | 2358.0 | |
| **4** | 1 | 0 | 0 | 0 | 6000 | 0.0 | |

In [125]:

```
X.tail()
```

Out[125]:

| | Gender | Married | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Property |
|---|---|---|---|---|---|---|---|
| **609** | 0 | 0 | 0 | 0 | 2900 | 0.0 | |
| **610** | 1 | 1 | 0 | 0 | 4106 | 0.0 | |
| **611** | 1 | 1 | 0 | 0 | 8072 | 240.0 | |
| **612** | 1 | 1 | 0 | 0 | 7583 | 0.0 | |
| **613** | 0 | 0 | 0 | 1 | 4583 | 0.0 | |

In [126]:

```
X.shape
```

Out[126]:

```
(614, 10)
```

In [127]:

```
Y = Loan_df[["Loan_Status"]] # Dependent Variable
```

In [128]:

```
Y
```

Out[128]:

| | Loan_Status |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 1 |
| **4** | 1 |
| **...** | ... |
| **609** | 1 |
| **610** | 1 |
| **611** | 1 |
| **612** | 1 |
| **613** | 0 |

614 rows × 1 columns

# Train and Test Split

In [129]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0
```

In [130]:

```
X_train.shape
```

Out[130]:

```
(491, 10)
```

In [131]:

```
X_test.shape
```

Out[131]:

```
(123, 10)
```

# Feature Scaling

In [132]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_Y = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
Y_train = sc_Y.fit_transform(Y_train)
Y_test = sc_Y.transform(Y_test)
```

In [133]:

```
X_train
```

Out[133]:

```
array([[ 0.37185886,  0.68308841, -0.53102197, ..., -0.18814516,
         0.27644707,  0.45859914],
       [ 0.37185886, -1.38725979, -0.53102197, ...,  0.56269516,
         0.27644707,  0.01148007],
       [ 0.37185886,  0.68308841, -0.53102197, ...,  0.01977985,
         0.27644707, -2.37482957],
       ...,
       [ 0.37185886,  0.68308841, -0.53102197, ...,  0.60890072,
         0.27644707,  0.45859914],
       [ 0.37185886,  0.68308841,  1.88316125, ...,  0.14684514,
         0.27644707,  0.45859914],
       [-1.968945  ,  0.68308841, -0.53102197, ...,  0.40097571,
         0.27644707,  0.45859914]])
```

In [134]:

```
X_test
```

Out[134]:

```
array([[ 0.37185886, -1.38725979, -0.53102197, ..., -0.73106046,
         0.27644707,  0.45859914],
       [-1.968945  , -1.38725979, -0.53102197, ..., -0.40762156,
         0.27644707,  0.45859914],
       [ 0.37185886,  0.68308841, -0.53102197, ...,  2.04127302,
         0.27644707,  0.45859914],
       ...,
       [ 0.37185886,  0.68308841, -0.53102197, ...,  3.33502864,
         0.27644707,  0.45859914],
       [ 0.37185886,  0.68308841,  1.88316125, ..., -0.89277992,
        -2.57314376, -2.37482957],
       [ 0.37185886,  0.68308841, -0.53102197, ..., -0.59244379,
         0.27644707,  0.45859914]])
```

In [135]:

```
Y_train
```

Out[135]:

```
array([[ 0.69203733],
       [-1.44500876],
       [-1.44500876],
       [ 0.69203733],
       [ 0.69203733],
       [-1.44500876],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [-1.44500876],
       [-1.44500876],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.692037331,
```

In [136]:

```
Y_test
```

Out[136]:

```
array([[ 0.69203733],
       [-1.44500876],
       [ 0.69203733],
       [-1.44500876],
       [ 0.69203733],
       [-1.44500876],
       [ 0.69203733],
       [ 0.69203733],
       [-1.44500876],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [ 0.69203733],
       [-1.44500876],
       [-1.44500876],
       [ 0.692037331,
```

In [137]:

```
from sklearn import preprocessing
from sklearn import utils

#convert y values to categorical values
value = preprocessing.LabelEncoder()
Y_train_transformed = value.fit_transform(Y_train)

#view transformed values
print(Y_train_transformed)
```

```
[1 0 0 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 1 1 1 0
 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1
 1 0 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 1 0 1 1 0 0 0 1
 1 1 0 1 0 0 1 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 0 1 0
 1 1 1 1 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 1 1 0 1
 1 1 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1
 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1
 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 1
 0 1 1 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0
 1 0 0 0 1 1 0 1 1 1 1 0 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1
 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 0
 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1 1 1 0 1 1 0 0
 0 0 1 1 1 1 0 1 0 1]

C:\Users\Lenovo\anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:
115: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using r
avel().
  y = column_or_1d(y, warn=True)
```

In [138]:

```
from sklearn import preprocessing
from sklearn import utils

#convert y values to categorical values
value = preprocessing.LabelEncoder()
Y_test_transformed = value.fit_transform(Y_test)

#view transformed values
print(Y_test_transformed)
```

```
[1 0 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1
 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1
 1 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1
 0 0 1 0 1 1 1 1 1 1 0 1]
```

# Using the logistics regression,Decision Tree,SVM,Random Forest algorithm to get the best possible accuracy

# 1. Logistics Regression

#Lets build our model using LogisticRegression from Scikit-learn package. This function implements logistic regression and can use different numerical optimizers to find parameters, including ‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’ solvers. You can find extensive information about the pros and cons of these optimizers if you search it in internet.

In [139]:

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(solver='saga')
LR.fit(X_train,Y_train_transformed) #training
LR
```

Out[139]:

```
LogisticRegression(solver='saga')
```

In [140]:

```
yhat = LR.predict(X_test)#only questions passed and answers are saved for evaluation
yhat[:5]
```

Out[140]:

```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [141]:

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob[:5]
```

Out[141]:

```
array([[0.22180067, 0.77819933],
       [0.20800595, 0.79199405],
       [0.21233114, 0.78766886],
       [0.15318661, 0.84681339],
       [0.16363189, 0.83636811]])
```

In [142]:

```
from sklearn.metrics import f1_score
f1_score(Y_test_transformed, yhat) #actualvale,predvalue
```

Out[142]:

```
0.9025641025641026
```

In [143]:

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(solver='newton-cg')
LR.fit(X_train,Y_train_transformed) #training
LR
```

Out[143]:

```
LogisticRegression(solver='newton-cg')
```

In [144]:

```
yhat = LR.predict(X_test)#only questions passed and answers are saved for evaluation
yhat[:5]
```

Out[144]:

```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [145]:

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob[:5]
```

Out[145]:

```
array([[0.22175263, 0.77824737],
       [0.2080085 , 0.7919915 ],
       [0.21242962, 0.78757038],
       [0.15315056, 0.84684944],
       [0.16361364, 0.83638636]])
```

In [146]:

```
from sklearn.metrics import f1_score
f1_score(Y_test_transformed, yhat) #actualvale,predvalue
```

Out[146]:

```
0.9025641025641026
```

In [147]:

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(solver='lbfgs')
LR.fit(X_train,Y_train_transformed) #training
LR
```

Out[147]:

```
LogisticRegression()
```

In [148]:

```
yhat = LR.predict(X_test)#only questions passed and answers are saved for evaluation
yhat[:5]
```

Out[148]:

```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [149]:

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob[:5]
```

Out[149]:

```
array([[0.22175951, 0.77824049],
       [0.20801907, 0.79198093],
       [0.21243086, 0.78756914],
       [0.15314754, 0.84685246],
       [0.16360924, 0.83639076]])
```

In [150]:

```
from sklearn.metrics import f1_score
f1_score(Y_test_transformed, yhat) # actualvale,predvalue
```

Out[150]:

```
0.9025641025641026
```

In [151]:

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(solver='liblinear')
LR.fit(X_train,Y_train_transformed) #training
LR
```

Out[151]:

```
LogisticRegression(solver='liblinear')
```

In [152]:

```
yhat = LR.predict(X_test)#only questions passed and answers are saved for evaluation
yhat[:5]
```

Out[152]:

```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [153]:

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob[:5]
```

Out[153]:

```
array([[0.22325182, 0.77674818],
       [0.20950513, 0.79049487],
       [0.21394965, 0.78605035],
       [0.15451167, 0.84548833],
       [0.16500183, 0.83499817]])
```

In [154]:

```
from sklearn.metrics import f1_score
f1_score(Y_test_transformed, yhat) # actualvale,predvalue
```

Out[154]:

```
0.9025641025641026
```

In [155]:

```
X_test
```

Out[155]:

```
array([[ 0.37185886, -1.38725979, -0.53102197, ..., -0.73106046,
         0.27644707,  0.45859914],
       [-1.968945  , -1.38725979, -0.53102197, ..., -0.40762156,
         0.27644707,  0.45859914],
       [ 0.37185886,  0.68308841, -0.53102197, ...,  2.04127302,
         0.27644707,  0.45859914],
       ...,
       [ 0.37185886,  0.68308841, -0.53102197, ...,  3.33502864,
         0.27644707,  0.45859914],
       [ 0.37185886,  0.68308841,  1.88316125, ..., -0.89277992,
        -2.57314376, -2.37482957],
       [ 0.37185886,  0.68308841, -0.53102197, ..., -0.59244379,
         0.27644707,  0.45859914]])
```

In [156]:

```
X_test[:,4]
```

Out[156]:

```
array([ 0.22037425, -0.20880471,  0.07726449, -0.19332119, -0.43759993,
       -0.45248214,  0.1199569 ,  1.09090851, -0.07441382, -0.53155329,
       -0.40618193, -0.40948909,  0.40798032, -0.09305416,  1.39216057,
       -0.26217021, -0.42843008, -0.49201772, -0.46886761, -0.47578258,
       -0.44707043, -0.48014201,  0.09485256,  0.40798032, -0.48645568,
        0.06704236, -0.29449017,  0.0133762 , -0.32906501, -0.09305416,
        0.52072435, -0.11034158, -0.45383507,  0.34303976, -0.1453674 ,
       -0.52163182, -0.48645568, -0.46135134,  0.09485256, -0.29223529,
       -0.35702553, -0.33357477, -0.64940839, -0.16009928, -0.26472575,
        0.40798032, -0.28096089, -0.16115156, -0.15573985, -0.24337954,
       -0.36769863, -0.13063551, -0.45383507, -0.26367347, -0.69059754,
        0.10732956, -0.38874418,  0.0897415 , -0.24247759, -0.2146674 ,
       -0.52343572,  0.65857273, -0.51892596, -0.31703898,  0.04750006,
        0.05050657, -0.53155329, -0.48901121, -0.28742488, -0.43895286,
       -0.55665763, -0.48044266, -0.24533377,  0.14866904, -0.29343789,
       -0.27134006, -0.56868366,  1.03438617,  0.61392609, -0.21842553,
       -0.74772119,  1.9112341 ,  0.27389008,  0.29523629, -0.38333247,
       -0.14942618, -0.42602488, -0.38273117, -0.29343789, -0.47022054,
       -0.55485373, -0.36860058, -0.20083746, -0.36860058, -0.40873746,
       -0.36363985,  0.20008032, -0.39806436, -0.35266609,  0.5891224 ,
        0.04569616, -0.57108887,  0.03006232,  0.06478748,  0.01262458,
       -0.10057043, -0.4563906 , -0.33207152, -0.49397195, -0.31523507,
        0.11740136, -0.33086891, -0.28096089, -0.36363985, -0.59393832,
       -0.47142314, -0.44376327, -0.53501078, -0.46886761,  0.470666  ,
        1.34751393, -0.40167217, -0.60040232])
```

In [157]:

```
X_test[:,5]
```

Out[157]:

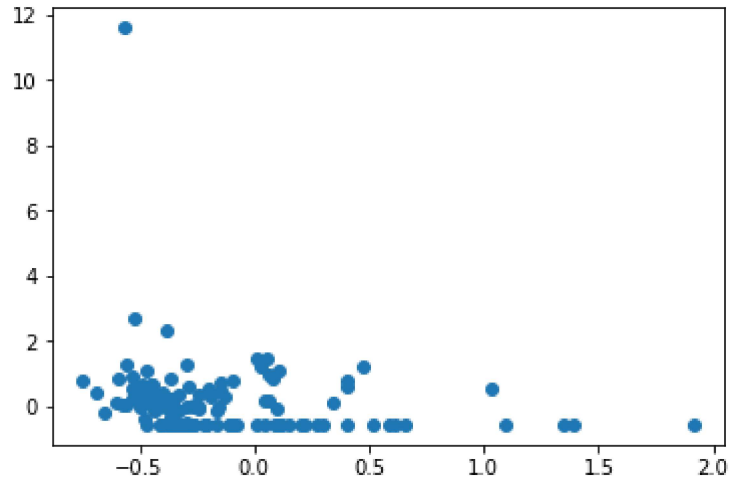
```
array([-5.47973128e-01, -5.47973128e-01,  8.54762572e-01,  3.32238137e-01,
       -1.28876210e-01,  6.90643213e-01, -5.47973128e-01, -5.47973128e-01,
       -5.47973128e-01,  5.83265340e-01, -5.47973128e-01, -5.47973128e-01,
       -5.47973128e-01,  7.68932098e-01, -5.47973128e-01, -5.47973128e-01,
        5.79503630e-03,  2.66241890e-02, -5.47973128e-01,  2.89861585e-01,
        6.87411103e-01,  2.92375448e-01, -8.11128079e-02,  5.89370436e-01,
        2.19832537e-01,  1.71710012e-01,  1.27062738e+00,  1.47209556e+00,
       -9.90689740e-02, -5.47973128e-01, -5.47973128e-01, -5.47973128e-01,
        3.49835180e-01,  1.10299923e-01,  7.38765739e-01,  6.14509069e-01,
        6.49344031e-01,  1.85356698e-01, -5.47973128e-01, -4.61065284e-01,
       -5.47973128e-01,  3.49835180e-01, -1.57965199e-01, -1.06610564e-01,
       -5.47973128e-01,  7.98739334e-01, -5.47973128e-01, -5.47973128e-01,
       -6.41515669e-03,  3.49835180e-01, -5.47973128e-01,  2.89861585e-01,
        6.25365213e-02,  4.09891220e-02,  4.47875847e-01,  1.09788906e+00,
       -2.40121995e-02, -5.47973128e-01,  4.90631295e-05, -5.47973128e-01,
        2.67695431e+00, -5.47973128e-01,  3.13922847e-01, -5.47973128e-01,
       -5.47973128e-01,  1.47209556e+00,  9.18327400e-01,  3.60968003e-01,
        5.89011313e-01,  4.52185327e-01,  1.27026826e+00, -3.66615850e-01,
       -4.12501190e-02, -5.47973128e-01, -2.40121995e-02, -1.53932398e-02,
        1.16036828e+01,  5.29396841e-01, -5.47973128e-01, -5.47973128e-01,
        7.74678071e-01, -5.47973128e-01, -5.47973128e-01, -5.47973128e-01,
       -5.47973128e-01,  4.78042206e-01,  1.76019491e-01,  2.36308053e+00,
       -5.47973128e-01,  1.89666178e-01,  4.24256153e-02, -5.47973128e-01,
        5.29396841e-01,  8.88520164e-01, -5.47973128e-01,  2.61490842e-01,
       -5.47973128e-01,  4.54340067e-01,  1.05631320e-01, -5.47973128e-01,
        1.89666178e-01,  6.93598645e-02,  1.24764349e+00,  1.00810823e+00,
       -5.47973128e-01, -5.47973128e-01,  2.30247113e-01, -5.47973128e-01,
       -3.54217010e-03, -5.47973128e-01, -5.47973128e-01, -5.47973128e-01,
       -5.47973128e-01, -2.96586802e-01,  8.49016599e-01,  3.37624987e-01,
        3.56022721e-02,  2.45330293e-01,  1.10399416e+00,  1.24620699e+00,
       -5.47973128e-01,  4.96080817e-02,  9.95262236e-02])
```

In [158]:

```
plt.scatter(X_test[:,4],X_test[:,5])
```

Out[158]:

```
<matplotlib.collections.PathCollection at 0x250590fa430>
```

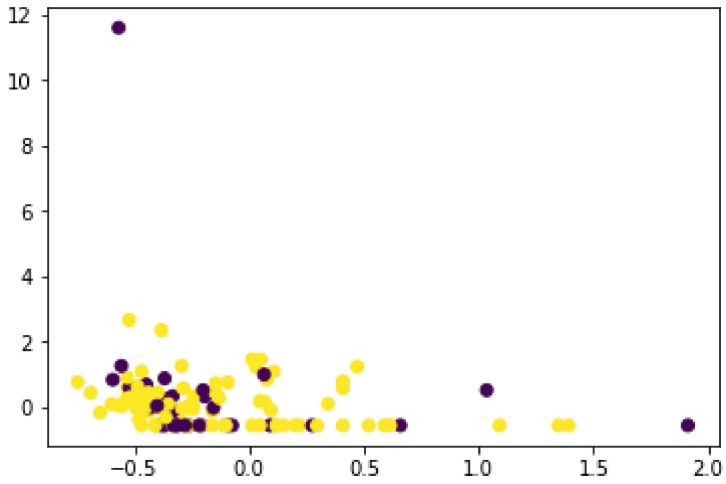


In [159]:

```
import matplotlib.pyplot as plt
plt.scatter(X_test[:,4],X_test[:,5],c = Y_test_transformed) #coloring based on actual value
```

Out[159]:

```
<matplotlib.collections.PathCollection at 0x250591e0b80>
```




# 2. Decision Tree

In [160]:

```
from sklearn.tree import DecisionTreeClassifier
Loan_df = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
Loan_df # it shows the default parameters
```

Out[160]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

In [161]:

```
Loan_df.fit(X_train,Y_train_transformed)
```

Out[161]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

# Prediction

In [162]:

```
y_pred = Loan_df.predict(X_test)
```

In [163]:

```
y_pred
```

Out[163]:

```
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1], dtype=int64)
```

In [164]:

```
print (y_pred [0:5])#predicted by the ml model
print (Y_test_transformed [0:5])#actual values we have
```

```
[1 1 1 1 1]
[1 0 1 0 1]
```

In [165]:

```
Y_test_transformed
```

Out[165]:

```
array([1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1], dtype=int64)
```

# Evaluation

In [166]:

```
from sklearn import metrics
print("DecisionTrees's Accuracy:", metrics.accuracy_score(Y_test_transformed, y_pred))
```

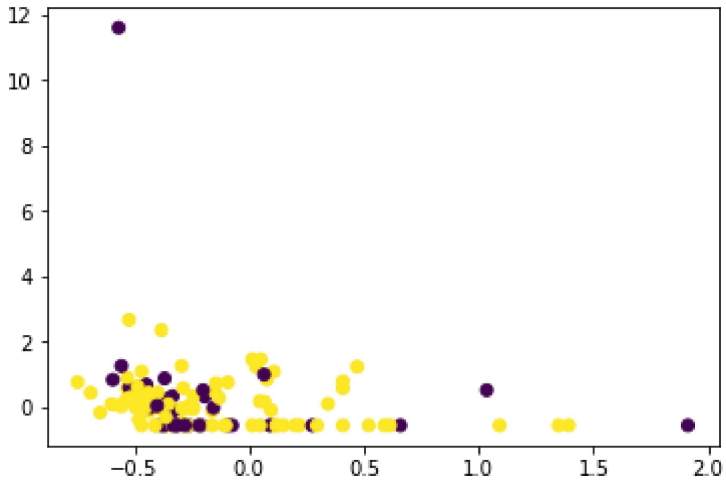
```
DecisionTrees's Accuracy: 0.8211382113821138
```

In [167]:

```
import matplotlib.pyplot as plt
plt.scatter(X_test[:,4],X_test[:,5],c = Y_test_transformed)
```

Out[167]:

```
<matplotlib.collections.PathCollection at 0x250593b3970>
```




# 3.SVM

In [168]:

```
from sklearn import svm
clf = svm.SVC(kernel='poly')
clf.fit(X_train, Y_train_transformed) # question and answers
```

Out[168]:

```
SVC(kernel='poly')
```

In [169]:

```
yhat = clf.predict(X_test) #question
yhat [0:5]
```

Out[169]:

```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [170]:

```
from sklearn.metrics import f1_score
f1_score(Y_test_transformed, yhat, average='weighted')
```

Out[170]:

```
0.8180189148830265
```

In [171]:

```
from sklearn import svm
clf2 = svm.SVC(kernel='rbf')
clf2.fit(X_train, Y_train_transformed)
yhat2 = clf2.predict(X_test)
print("Avg F1-score: %.4f" % f1_score(Y_test_transformed, yhat2, average="weighted"))
```

```
Avg F1-score: 0.8070
```

In [172]:

```
from sklearn import svm
clf2 = svm.SVC(kernel='linear')
clf2.fit(X_train, Y_train_transformed)
yhat2 = clf2.predict(X_test)
print("Avg F1-score: %.4f" % f1_score(Y_test_transformed, yhat2, average="weighted"))
```

```
Avg F1-score: 0.8070
```

# Random Forest

In [173]:

```
# Fitting the classifier into the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 400, criterion = 'entropy')#5000 decisio
classifier.fit(X_train,Y_train_transformed)#question,answers (300,2) (300,1)
```

Out[173]:

```
RandomForestClassifier(criterion='entropy', n_estimators=400)
```

In [174]:

```
# Predicting the test set results
Y_Pred = classifier.predict(X_test)#examination only question are given
```

In [175]:

```
from sklearn.metrics import f1_score
f1_score(Y_test_transformed,Y_Pred)#actual,predict
```

Out[175]:

```
0.8749999999999999
```

# Logistic Regression : 90.25%

# Decision Tree : 82.11%

# SVM : 81.80%

# Random Forest : 88.08%

# Ans : Logistic Regression gives the best possible accuracy that is 90.25%

# Project 2

#ApplyExploratory Data Analysis on the FifaDataset Convert the appropriate columns to machine understandable columns using label encoding Handle Null values Apply the correlation and remove the column which are more than 60% correlated Apply the variance and remove the columns whose variance is less that 20% Understand the Dataset whether the dataset is corr

In [5]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [6]:

```
Fifa_df = pd.read_excel("C:/Users/Lenovo/Documents/Data Set/Fifa.xlsx")
```

In [7]:

```
Fifa_df
```

Out[7]:

| | Unnamed: 0 | ID | Name | Age | Photo | Nation |
|---|---|---|---|---|---|---|
| **0** | 0 | 158023 | L. Messi | 31 | https://cdn.sofifa.org/players/4/19/158023.png | Arge |
| **1** | 1 | 20801 | Cristiano Ronaldo | 33 | https://cdn.sofifa.org/players/4/19/20801.png | Port |
| **2** | 2 | 190871 | Neymar Jr | 26 | https://cdn.sofifa.org/players/4/19/190871.png | B |
| **3** | 3 | 193080 | De Gea | 27 | https://cdn.sofifa.org/players/4/19/193080.png | S |
| **4** | 4 | 192985 | K. De Bruyne | 27 | https://cdn.sofifa.org/players/4/19/192985.png | Bel |
| **...** | ... | ... | ... | ... | ... | |
| **18202** | 18202 | 238813 | J. Lundstram | 19 | https://cdn.sofifa.org/players/4/19/238813.png | Eng |
| **18203** | 18203 | 243165 | N. Christoffersson | 19 | https://cdn.sofifa.org/players/4/19/243165.png | Swe |
| **18204** | 18204 | 241638 | B. Worman | 16 | https://cdn.sofifa.org/players/4/19/241638.png | Eng |
| **18205** | 18205 | 246268 | D. Walker-Rice | 17 | https://cdn.sofifa.org/players/4/19/246268.png | Eng |
| **18206** | 18206 | 246269 | G. Nugent | 16 | https://cdn.sofifa.org/players/4/19/246269.png | Eng |

18207 rows × 89 columns

In [8]:

```
Fifa_df.head()
```

Out[8]:

| | Unnamed: 0 | ID | Name | Age | Photo | Nationality | |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 158023 | L. Messi | 31 | https://cdn.sofifa.org/players/4/19/158023.png | Argentina | https |
| **1** | 1 | 20801 | Cristiano Ronaldo | 33 | https://cdn.sofifa.org/players/4/19/20801.png | Portugal | https |
| **2** | 2 | 190871 | Neymar Jr | 26 | https://cdn.sofifa.org/players/4/19/190871.png | Brazil | https |
| **3** | 3 | 193080 | De Gea | 27 | https://cdn.sofifa.org/players/4/19/193080.png | Spain | https |
| **4** | 4 | 192985 | K. De Bruyne | 27 | https://cdn.sofifa.org/players/4/19/192985.png | Belgium | http |

5 rows × 89 columns

In [9]:

```
Fifa_df.tail()
```

Out[9]:

| | Unnamed: 0 | ID | Name | Age | Photo | Nation |
|---|---|---|---|---|---|---|
| **18202** | 18202 | 238813 | J. Lundstram | 19 | https://cdn.sofifa.org/players/4/19/238813.png | Eng |
| **18203** | 18203 | 243165 | N. Christoffersson | 19 | https://cdn.sofifa.org/players/4/19/243165.png | Swe |
| **18204** | 18204 | 241638 | B. Worman | 16 | https://cdn.sofifa.org/players/4/19/241638.png | Eng |
| **18205** | 18205 | 246268 | D. Walker-Rice | 17 | https://cdn.sofifa.org/players/4/19/246268.png | Eng |
| **18206** | 18206 | 246269 | G. Nugent | 16 | https://cdn.sofifa.org/players/4/19/246269.png | Eng |

5 rows × 89 columns

In [10]:

```
Fifa_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 89 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                18207 non-null  int64
 1   ID                        18207 non-null  int64
 2   Name                      18207 non-null  object
 3   Age                       18207 non-null  int64
 4   Photo                     18207 non-null  object
 5   Nationality               18207 non-null  object
 6   Flag                      18207 non-null  object
 7   Overall                   18207 non-null  int64
 8   Potential                 18207 non-null  int64
 9   Club                      17966 non-null  object
 10  Club Logo                 18207 non-null  object
 11  Value                     18207 non-null  object
 12  Wage                      18207 non-null  object
 13  Special                   18207 non-null  int64
 14  Preferred Foot            18159 non-null  object
 15  International Reputation  18159 non-null  float64
 16  Weak Foot                 18159 non-null  float64
 17  Skill Moves               18159 non-null  float64
 18  Work Rate                 18159 non-null  object
 19  Body Type                 18159 non-null  object
 20  Real Face                 18159 non-null  object
 21  Position                  18147 non-null  object
 22  Jersey Number             18147 non-null  float64
 23  Joined                    16654 non-null  datetime64[ns]
 24  Loaned From               1264 non-null   object
 25  Contract Valid Until      17918 non-null  object
 26  Height                    18159 non-null  object
 27  Weight                    18159 non-null  object
 28  LS                        16122 non-null  object
 29  ST                        16122 non-null  object
 30  RS                        16122 non-null  object
 31  LW                        16122 non-null  object
 32  LF                        16122 non-null  object
 33  CF                        16122 non-null  object
 34  RF                        16122 non-null  object
 35  RW                        16122 non-null  object
 36  LAM                       16122 non-null  object
 37  CAM                       16122 non-null  object
 38  RAM                       16122 non-null  object
 39  LM                        16122 non-null  object
 40  LCM                       16122 non-null  object
 41  CM                        16122 non-null  object
 42  RCM                       16122 non-null  object
 43  RM                        16122 non-null  object
 44  LWB                       16122 non-null  object
 45  LDM                       16122 non-null  object
 46  CDM                       16122 non-null  object
 47  RDM                       16122 non-null  object
 48  RWB                       16122 non-null  object
 49  LB                        16122 non-null  object
 50  LCB                       16122 non-null  object
```

```
 51  CB                        16122 non-null  object
 52  RCB                       16122 non-null  object
 53  RB                        16122 non-null  object
 54  Crossing                  18159 non-null  float64
 55  Finishing                 18159 non-null  float64
 56  HeadingAccuracy           18159 non-null  float64
 57  ShortPassing              18159 non-null  float64
 58  Volleys                   18159 non-null  float64
 59  Dribbling                 18159 non-null  float64
 60  Curve                     18159 non-null  float64
 61  FKAccuracy                18159 non-null  float64
 62  LongPassing               18159 non-null  float64
 63  BallControl               18159 non-null  float64
 64  Acceleration              18159 non-null  float64
 65  SprintSpeed               18159 non-null  float64
 66  Agility                   18159 non-null  float64
 67  Reactions                 18159 non-null  float64
 68  Balance                   18159 non-null  float64
 69  ShotPower                 18159 non-null  float64
 70  Jumping                   18159 non-null  float64
 71  Stamina                   18159 non-null  float64
 72  Strength                  18159 non-null  float64
 73  LongShots                 18159 non-null  float64
 74  Aggression                18159 non-null  float64
 75  Interceptions             18159 non-null  float64
 76  Positioning               18159 non-null  float64
 77  Vision                    18159 non-null  float64
 78  Penalties                 18159 non-null  float64
 79  Composure                 18159 non-null  float64
 80  Marking                   18159 non-null  float64
 81  StandingTackle            18159 non-null  float64
 82  SlidingTackle             18159 non-null  float64
 83  GKDiving                  18159 non-null  float64
 84  GKHandling                18159 non-null  float64
 85  GKKicking                 18159 non-null  float64
 86  GKPositioning             18159 non-null  float64
 87  GKReflexes                18159 non-null  float64
 88  Release Clause            16643 non-null  object
dtypes: datetime64[ns](1), float64(38), int64(6), object(44)
memory usage: 12.4+ MB
```

In [11]:

```
set(Fifa_df["Crossing"])
```

Out[11]:

```
{nan,
 nan,
 nan,
 nan,
 nan,
 5.0,
 6.0,
 7.0,
 8.0,
 9.0,
 10.0,
 11.0,
 12.0,
 13.0,
 14.0,
 15.0,
 16.0,
 17.0.
```

In [12]:

```
Fifa_df.describe()
```

Out[12]:

| | Unnamed: 0 | ID | Age | Overall | Potential | Special | |
|---|---|---|---|---|---|---|---|
| **count** | 18207.000000 | 18207.000000 | 18207.000000 | 18207.000000 | 18207.000000 | 18207.000000 | 1 |
| **mean** | 9103.000000 | 214298.338606 | 25.122206 | 66.238699 | 71.307299 | 1597.809908 | |
| **std** | 5256.052511 | 29965.244204 | 4.669943 | 6.908930 | 6.136496 | 272.586016 | |
| **min** | 0.000000 | 16.000000 | 16.000000 | 46.000000 | 48.000000 | 731.000000 | |
| **25%** | 4551.500000 | 200315.500000 | 21.000000 | 62.000000 | 67.000000 | 1457.000000 | |
| **50%** | 9103.000000 | 221759.000000 | 25.000000 | 66.000000 | 71.000000 | 1635.000000 | |
| **75%** | 13654.500000 | 236529.500000 | 28.000000 | 71.000000 | 75.000000 | 1787.000000 | |
| **max** | 18206.000000 | 246620.000000 | 45.000000 | 94.000000 | 95.000000 | 2346.000000 | |

8 rows × 44 columns

In [13]:

```
Fifa_df.shape
```

Out[13]:

```
(18207, 89)
```

In [14]:

```
Fifa_df.isnull().sum()
```

Out[14]:

```
Unnamed: 0           0
ID                   0
Name                 0
Age                  0
Photo                0
                  ...
GKHandling          48
GKKicking           48
GKPositioning       48
GKReflexes          48
Release Clause    1564
Length: 89, dtype: int64
```

In [15]:

```
Fifa_df
```

Out[15]:

| | Unnamed: 0 | ID | Name | Age | Photo | Nation |
|---|---|---|---|---|---|---|
| **0** | 0 | 158023 | L. Messi | 31 | https://cdn.sofifa.org/players/4/19/158023.png | Arge |
| **1** | 1 | 20801 | Cristiano Ronaldo | 33 | https://cdn.sofifa.org/players/4/19/20801.png | Port |
| **2** | 2 | 190871 | Neymar Jr | 26 | https://cdn.sofifa.org/players/4/19/190871.png | B |
| **3** | 3 | 193080 | De Gea | 27 | https://cdn.sofifa.org/players/4/19/193080.png | S |
| **4** | 4 | 192985 | K. De Bruyne | 27 | https://cdn.sofifa.org/players/4/19/192985.png | Bel |
| **...** | ... | ... | ... | ... | ... | |
| **18202** | 18202 | 238813 | J. Lundstram | 19 | https://cdn.sofifa.org/players/4/19/238813.png | Eng |
| **18203** | 18203 | 243165 | N. Christoffersson | 19 | https://cdn.sofifa.org/players/4/19/243165.png | Swe |
| **18204** | 18204 | 241638 | B. Worman | 16 | https://cdn.sofifa.org/players/4/19/241638.png | Eng |
| **18205** | 18205 | 246268 | D. Walker-Rice | 17 | https://cdn.sofifa.org/players/4/19/246268.png | Eng |
| **18206** | 18206 | 246269 | G. Nugent | 16 | https://cdn.sofifa.org/players/4/19/246269.png | Eng |

18207 rows × 89 columns

# Apply Label Encoding

In [16]:

```
from sklearn.preprocessing import LabelEncoder
```

In [17]:

```
le_Height = LabelEncoder()
print(le_Height)
Fifa_df["Height"] = le_Height.fit_transform(Fifa_df["Height"])
Fifa_df

le_Weight = LabelEncoder()
print(le_Weight)
Fifa_df["Weight"] = le_Weight.fit_transform(Fifa_df["Weight"])
Fifa_df


le_LS = LabelEncoder()
print(le_LS)
Fifa_df["LS"] = le_LS.fit_transform(Fifa_df["LS"])
Fifa_df


le_ST = LabelEncoder()
print(le_ST)
Fifa_df["ST"] = le_ST.fit_transform(Fifa_df["ST"])
Fifa_df


le_RS = LabelEncoder()
print(le_RS)
Fifa_df["RS"] = le_RS.fit_transform(Fifa_df["RS"])
Fifa_df



le_LW = LabelEncoder()
print(le_LW)
Fifa_df["LW"] = le_LW.fit_transform(Fifa_df["LW"])
Fifa_df


le_LF = LabelEncoder()
print(le_LF)
Fifa_df["LF"] = le_LF.fit_transform(Fifa_df["LF"])
Fifa_df


le_CF = LabelEncoder()
print(le_CF)
Fifa_df["CF"] = le_CF.fit_transform(Fifa_df["CF"])
Fifa_df


le_RF = LabelEncoder()
print(le_RF)
Fifa_df["RF"] = le_RF.fit_transform(Fifa_df["RF"])
Fifa_df


le_RW = LabelEncoder()
print(le_RW)
Fifa_df["RW"] = le_RW.fit_transform(Fifa_df["RW"])
Fifa_df
```

```
le_LAM = LabelEncoder()
print(le_LAM)
Fifa_df["LAM"] = le_LAM.fit_transform(Fifa_df["LAM"])
Fifa_df


le_CAM = LabelEncoder()
print(le_CAM)
Fifa_df["CAM"] = le_RW.fit_transform(Fifa_df["CAM"])
Fifa_df


le_RAM = LabelEncoder()
print(le_RAM)
Fifa_df["RAM"] = le_RAM.fit_transform(Fifa_df["RAM"])
Fifa_df

le_LM = LabelEncoder()
print(le_LM)
Fifa_df["LM"] = le_LM.fit_transform(Fifa_df["LM"])
Fifa_df

le_LCM = LabelEncoder()
print(le_LCM)
Fifa_df["LCM"] = le_LCM.fit_transform(Fifa_df["LCM"])
Fifa_df


le_CM = LabelEncoder()
print(le_CM)
Fifa_df["CM"] = le_CM.fit_transform(Fifa_df["CM"])
Fifa_df


le_RCM = LabelEncoder()
print(le_RCM)
Fifa_df["RCM"] = le_RCM.fit_transform(Fifa_df["RCM"])
Fifa_df

le_RM = LabelEncoder()
print(le_RM)
Fifa_df["RM"] = le_RM.fit_transform(Fifa_df["RM"])
Fifa_df

le_LWB = LabelEncoder()
print(le_LWB)
Fifa_df["LWB"] = le_LWB.fit_transform(Fifa_df["LWB"])
Fifa_df

le_LDM= LabelEncoder()
print(le_LDM)
Fifa_df["LDM"] = le_LDM.fit_transform(Fifa_df["LDM"])
Fifa_df

le_CDM= LabelEncoder()
print(le_CDM)
Fifa_df["CDM"] = le_CDM.fit_transform(Fifa_df["CDM"])
Fifa_df


le_RDM= LabelEncoder()
```

```
print(le_RDM)
Fifa_df["RDM"] = le_RDM.fit_transform(Fifa_df["RDM"])
Fifa_df

le_RWB= LabelEncoder()
print(le_RWB)
Fifa_df["RWB"] = le_RWB.fit_transform(Fifa_df["RWB"])
Fifa_df

le_LB= LabelEncoder()
print(le_LB)
Fifa_df["LB"] = le_LB.fit_transform(Fifa_df["LB"])
Fifa_df

le_LCB= LabelEncoder()
print(le_LCB)
Fifa_df["LCB"] = le_LCB.fit_transform(Fifa_df["LCB"])
Fifa_df

le_CB= LabelEncoder()
print(le_CB)
Fifa_df["CB"] = le_CB.fit_transform(Fifa_df["CB"])
Fifa_df

le_RCB= LabelEncoder()
print(le_RCB)
Fifa_df["RCB"] = le_RCB.fit_transform(Fifa_df["RCB"])
Fifa_df

le_RB= LabelEncoder()
print(le_RB)
Fifa_df["RB"] = le_RB.fit_transform(Fifa_df["RB"])
Fifa_df
```

```
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
LabelEncoder()
```

```
LabelEncoder()
LabelEncoder()
LabelEncoder()
```

Out[17]:

| | Unnamed: 0 | ID | Name | Age | Photo | Nation |
|---|---|---|---|---|---|---|
| **0** | 0 | 158023 | L. Messi | 31 | https://cdn.sofifa.org/players/4/19/158023.png | Arge |
| **1** | 1 | 20801 | Cristiano Ronaldo | 33 | https://cdn.sofifa.org/players/4/19/20801.png | Port |
| **2** | 2 | 190871 | Neymar Jr | 26 | https://cdn.sofifa.org/players/4/19/190871.png | B |
| **3** | 3 | 193080 | De Gea | 27 | https://cdn.sofifa.org/players/4/19/193080.png | S |
| **4** | 4 | 192985 | K. De Bruyne | 27 | https://cdn.sofifa.org/players/4/19/192985.png | Bel |
| **...** | ... | ... | ... | ... | ... | |
| **18202** | 18202 | 238813 | J. Lundstram | 19 | https://cdn.sofifa.org/players/4/19/238813.png | Eng |
| **18203** | 18203 | 243165 | N. Christoffersson | 19 | https://cdn.sofifa.org/players/4/19/243165.png | Swe |
| **18204** | 18204 | 241638 | B. Worman | 16 | https://cdn.sofifa.org/players/4/19/241638.png | Eng |
| **18205** | 18205 | 246268 | D. Walker-Rice | 17 | https://cdn.sofifa.org/players/4/19/246268.png | Eng |
| **18206** | 18206 | 246269 | G. Nugent | 16 | https://cdn.sofifa.org/players/4/19/246269.png | Eng |

18207 rows × 89 columns

In [18]:

```
Fifa_df1 = Fifa_df.drop(["Unnamed: 0","ID","Name","Age","Photo","Nationality","Flag","Club"
```

In [19]:

```
Fifa_df1
```

Out[19]:

| | Overall | Potential | Special | Height | Weight | LS | ST | RS | LW | LF | ... | Penalties | Comp |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 94 | 94 | 2202 | 8 | 21 | 91 | 91 | 91 | 104 | 101 | ... | 75.0 | |
| **1** | 94 | 94 | 2228 | 13 | 32 | 92 | 92 | 92 | 103 | 100 | ... | 85.0 | |
| **2** | 92 | 93 | 2143 | 10 | 17 | 86 | 86 | 86 | 103 | 99 | ... | 81.0 | |
| **3** | 91 | 93 | 1471 | 15 | 25 | 93 | 93 | 93 | 105 | 102 | ... | 40.0 | |
| **4** | 91 | 92 | 2281 | 2 | 19 | 83 | 83 | 83 | 102 | 96 | ... | 79.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **18202** | 47 | 65 | 1307 | 10 | 10 | 11 | 11 | 11 | 19 | 17 | ... | 43.0 | |
| **18203** | 47 | 63 | 1098 | 14 | 26 | 14 | 14 | 14 | 13 | 14 | ... | 43.0 | |
| **18204** | 47 | 67 | 1189 | 9 | 16 | 14 | 14 | 14 | 21 | 19 | ... | 55.0 | |
| **18205** | 47 | 66 | 1228 | 1 | 19 | 16 | 16 | 16 | 23 | 19 | ... | 50.0 | |
| **18206** | 46 | 66 | 1321 | 1 | 29 | 12 | 12 | 12 | 21 | 17 | ... | 33.0 | |

18207 rows × 65 columns

In [20]:

```
Fifa_df1.shape
```

Out[20]:

```
(18207, 65)
```

In [21]:

```
Fifa_df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 65 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Overall          18207 non-null  int64  
 1   Potential        18207 non-null  int64  
 2   Special          18207 non-null  int64  
 3   Height           18207 non-null  int32  
 4   Weight           18207 non-null  int32  
 5   LS               18207 non-null  int32  
 6   ST               18207 non-null  int32  
 7   RS               18207 non-null  int32  
 8   LW               18207 non-null  int32  
 9   LF               18207 non-null  int32  
 10  CF               18207 non-null  int32  
 11  RF               18207 non-null  int32  
 12  RW               18207 non-null  int32  
 13  LAM              18207 non-null  int32  
 14  CAM              18207 non-null  int32  
 15  RAM              18207 non-null  int32  
 16  LM               18207 non-null  int32  
 17  LCM              18207 non-null  int32  
 18  CM               18207 non-null  int32  
 19  RCM              18207 non-null  int32  
 20  RM               18207 non-null  int32  
 21  LWB              18207 non-null  int32  
 22  LDM              18207 non-null  int32  
 23  CDM              18207 non-null  int32  
 24  RDM              18207 non-null  int32  
 25  RWB              18207 non-null  int32  
 26  LB               18207 non-null  int32  
 27  LCB              18207 non-null  int32  
 28  CB               18207 non-null  int32  
 29  RCB              18207 non-null  int32  
 30  RB               18207 non-null  int32  
 31  Crossing         18159 non-null  float64
 32  Finishing        18159 non-null  float64
 33  HeadingAccuracy  18159 non-null  float64
 34  ShortPassing     18159 non-null  float64
 35  Volleys          18159 non-null  float64
 36  Dribbling        18159 non-null  float64
 37  Curve            18159 non-null  float64
 38  FKAccuracy       18159 non-null  float64
 39  LongPassing      18159 non-null  float64
 40  BallControl      18159 non-null  float64
 41  Acceleration     18159 non-null  float64
 42  SprintSpeed      18159 non-null  float64
 43  Agility          18159 non-null  float64
 44  Reactions        18159 non-null  float64
 45  Balance          18159 non-null  float64
 46  ShotPower        18159 non-null  float64
 47  Jumping          18159 non-null  float64
 48  Stamina          18159 non-null  float64
 49  Strength         18159 non-null  float64
 50  LongShots        18159 non-null  float64
```

```
 51  Aggression       18159 non-null  float64
 52  Interceptions    18159 non-null  float64
 53  Positioning      18159 non-null  float64
 54  Vision           18159 non-null  float64
 55  Penalties        18159 non-null  float64
 56  Composure        18159 non-null  float64
 57  Marking          18159 non-null  float64
 58  StandingTackle   18159 non-null  float64
 59  SlidingTackle    18159 non-null  float64
 60  GKDiving         18159 non-null  float64
 61  GKHandling       18159 non-null  float64
 62  GKKicking        18159 non-null  float64
 63  GKPositioning    18159 non-null  float64
 64  GKReflexes       18159 non-null  float64
dtypes: float64(34), int32(28), int64(3)
memory usage: 7.1 MB
```

In [22]:

```
Fifa_df1.columns
```

Out[22]:

```
Index(['Overall', 'Potential', 'Special', 'Height', 'Weight', 'LS', 'ST', 'R
S',
       'LW', 'LF', 'CF', 'RF', 'RW', 'LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM',
       'RCM', 'RM', 'LWB', 'LDM', 'CDM', 'RDM', 'RWB', 'LB', 'LCB', 'CB',
       'RCB', 'RB', 'Crossing', 'Finishing', 'HeadingAccuracy', 'ShortPassin
g',
       'Volleys', 'Dribbling', 'Curve', 'FKAccuracy', 'LongPassing',
       'BallControl', 'Acceleration', 'SprintSpeed', 'Agility', 'Reactions',
       'Balance', 'ShotPower', 'Jumping', 'Stamina', 'Strength', 'LongShot
s',
       'Aggression', 'Interceptions', 'Positioning', 'Vision', 'Penalties',
       'Composure', 'Marking', 'StandingTackle', 'SlidingTackle', 'GKDivin
g',
       'GKHandling', 'GKKicking', 'GKPositioning', 'GKReflexes'],
      dtype='object')
```

# Handling the Null Values

In [23]:

```
Fifa_df2 = Fifa_df1.dropna()
Fifa_df2.isnull().sum()
```

Out[23]:

```
Overall           0
Potential         0
Special           0
Height            0
Weight            0
                 ..
GKDiving          0
GKHandling        0
GKKicking         0
GKPositioning     0
GKReflexes        0
Length: 65, dtype: int64
```

In [24]:

```
Fifa_df2.shape
```

Out[24]:

```
(18159, 65)
```

In [25]:

```
Fifa_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18159 entries, 0 to 18206
Data columns (total 65 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Overall          18159 non-null  int64  
 1   Potential        18159 non-null  int64  
 2   Special          18159 non-null  int64  
 3   Height           18159 non-null  int32  
 4   Weight           18159 non-null  int32  
 5   LS               18159 non-null  int32  
 6   ST               18159 non-null  int32  
 7   RS               18159 non-null  int32  
 8   LW               18159 non-null  int32  
 9   LF               18159 non-null  int32  
 10  CF               18159 non-null  int32  
 11  RF               18159 non-null  int32  
 12  RW               18159 non-null  int32  
 13  LAM              18159 non-null  int32  
 14  CAM              18159 non-null  int32  
 15  RAM              18159 non-null  int32  
 16  LM               18159 non-null  int32  
 17  LCM              18159 non-null  int32  
 18  CM               18159 non-null  int32  
 19  RCM              18159 non-null  int32  
 20  RM               18159 non-null  int32  
 21  LWB              18159 non-null  int32  
 22  LDM              18159 non-null  int32  
 23  CDM              18159 non-null  int32  
 24  RDM              18159 non-null  int32  
 25  RWB              18159 non-null  int32  
 26  LB               18159 non-null  int32  
 27  LCB              18159 non-null  int32  
 28  CB               18159 non-null  int32  
 29  RCB              18159 non-null  int32  
 30  RB               18159 non-null  int32  
 31  Crossing         18159 non-null  float64
 32  Finishing        18159 non-null  float64
 33  HeadingAccuracy  18159 non-null  float64
 34  ShortPassing     18159 non-null  float64
 35  Volleys          18159 non-null  float64
 36  Dribbling        18159 non-null  float64
 37  Curve            18159 non-null  float64
 38  FKAccuracy       18159 non-null  float64
 39  LongPassing      18159 non-null  float64
 40  BallControl      18159 non-null  float64
 41  Acceleration     18159 non-null  float64
 42  SprintSpeed      18159 non-null  float64
 43  Agility          18159 non-null  float64
 44  Reactions        18159 non-null  float64
 45  Balance          18159 non-null  float64
 46  ShotPower        18159 non-null  float64
 47  Jumping          18159 non-null  float64
 48  Stamina          18159 non-null  float64
 49  Strength         18159 non-null  float64
 50  LongShots        18159 non-null  float64
```

```
 51  Aggression        18159 non-null  float64
 52  Interceptions     18159 non-null  float64
 53  Positioning       18159 non-null  float64
 54  Vision            18159 non-null  float64
 55  Penalties         18159 non-null  float64
 56  Composure         18159 non-null  float64
 57  Marking           18159 non-null  float64
 58  StandingTackle    18159 non-null  float64
 59  SlidingTackle     18159 non-null  float64
 60  GKDiving          18159 non-null  float64
 61  GKHandling        18159 non-null  float64
 62  GKKicking         18159 non-null  float64
 63  GKPositioning     18159 non-null  float64
 64  GKReflexes        18159 non-null  float64
dtypes: float64(34), int32(28), int64(3)
memory usage: 7.2 MB
```

# Apply the correlation and remove the column which are more than 60% correlated

In [26]:

```
X = Fifa_df2.drop("GKReflexes",axis=1)   #independent variable : all column except Target D
Y = Fifa_df2["GKReflexes"] #dependent variables only target column will be in Y
```

In [27]:

```
X
```

Out[27]:

| | Overall | Potential | Special | Height | Weight | LS | ST | RS | LW | LF | ... | Vision | Penalties |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 94 | 94 | 2202 | 8 | 21 | 91 | 91 | 91 | 104 | 101 | ... | 94.0 | 75.0 |
| **1** | 94 | 94 | 2228 | 13 | 32 | 92 | 92 | 92 | 103 | 100 | ... | 82.0 | 85.0 |
| **2** | 92 | 93 | 2143 | 10 | 17 | 86 | 86 | 86 | 103 | 99 | ... | 87.0 | 81.0 |
| **3** | 91 | 93 | 1471 | 15 | 25 | 93 | 93 | 93 | 105 | 102 | ... | 68.0 | 40.0 |
| **4** | 91 | 92 | 2281 | 2 | 19 | 83 | 83 | 83 | 102 | 96 | ... | 94.0 | 79.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. |
| **18202** | 47 | 65 | 1307 | 10 | 10 | 11 | 11 | 11 | 19 | 17 | ... | 52.0 | 43.0 |
| **18203** | 47 | 63 | 1098 | 14 | 26 | 14 | 14 | 14 | 13 | 14 | ... | 33.0 | 43.0 |
| **18204** | 47 | 67 | 1189 | 9 | 16 | 14 | 14 | 14 | 21 | 19 | ... | 43.0 | 55.0 |
| **18205** | 47 | 66 | 1228 | 1 | 19 | 16 | 16 | 16 | 23 | 19 | ... | 47.0 | 50.0 |
| **18206** | 46 | 66 | 1321 | 1 | 29 | 12 | 12 | 12 | 21 | 17 | ... | 49.0 | 33.0 |

18159 rows × 64 columns

In [28]:

```
Y
```

Out[28]:

```
0         8.0
1        11.0
2        11.0
3        94.0
4        13.0
         ...
18202     9.0
18203    12.0
18204    13.0
18205     9.0
18206     9.0
Name: GKReflexes, Length: 18159, dtype: float64
```

In [29]:

```
X.head()  # Independent Variable
```

Out[29]:

| | Overall | Potential | Special | Height | Weight | LS | ST | RS | LW | LF | ... | Vision | Penalties | Cc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 94 | 94 | 2202 | 8 | 21 | 91 | 91 | 91 | 104 | 101 | ... | 94.0 | 75.0 | |
| **1** | 94 | 94 | 2228 | 13 | 32 | 92 | 92 | 92 | 103 | 100 | ... | 82.0 | 85.0 | |
| **2** | 92 | 93 | 2143 | 10 | 17 | 86 | 86 | 86 | 103 | 99 | ... | 87.0 | 81.0 | |
| **3** | 91 | 93 | 1471 | 15 | 25 | 93 | 93 | 93 | 105 | 102 | ... | 68.0 | 40.0 | |
| **4** | 91 | 92 | 2281 | 2 | 19 | 83 | 83 | 83 | 102 | 96 | ... | 94.0 | 79.0 | |

5 rows × 64 columns

In [31]:

```
X.tail()
```

Out[31]:

| | Overall | Potential | Special | Height | Weight | LS | ST | RS | LW | LF | ... | Vision | Penalties |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **18202** | 47 | 65 | 1307 | 10 | 10 | 11 | 11 | 11 | 19 | 17 | ... | 52.0 | 43.0 |
| **18203** | 47 | 63 | 1098 | 14 | 26 | 14 | 14 | 14 | 13 | 14 | ... | 33.0 | 43.0 |
| **18204** | 47 | 67 | 1189 | 9 | 16 | 14 | 14 | 14 | 21 | 19 | ... | 43.0 | 55.0 |
| **18205** | 47 | 66 | 1228 | 1 | 19 | 16 | 16 | 16 | 23 | 19 | ... | 47.0 | 50.0 |
| **18206** | 46 | 66 | 1321 | 1 | 29 | 12 | 12 | 12 | 21 | 17 | ... | 49.0 | 33.0 |

5 rows × 64 columns

In [30]:

```
Y.head() # Dependent Variable
```

Out[30]:

```
0     8.0
1    11.0
2    11.0
3    94.0
4    13.0
Name: GKReflexes, dtype: float64
```

In [32]:

```
Y.tail()
```

Out[32]:

```
18202     9.0
18203    12.0
18204    13.0
18205     9.0
18206     9.0
Name: GKReflexes, dtype: float64
```

In [33]:

```
# separate dataset into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
    X,
    Y,
    test_size=0.3,
    random_state=40)

X_train.shape, X_test.shape
```

Out[33]:

```
((12711, 64), (5448, 64))
```

In [34]:

```
X_train
```

Out[34]:

| | Overall | Potential | Special | Height | Weight | LS | ST | RS | LW | LF | ... | Vision | Penalties |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **12383** | 63 | 65 | 1732 | 2 | 23 | 34 | 34 | 34 | 56 | 47 | ... | 69.0 | 62.0 |
| **8471** | 67 | 67 | 1739 | 13 | 35 | 28 | 28 | 28 | 40 | 35 | ... | 52.0 | 52.0 |
| **7381** | 68 | 68 | 1746 | 11 | 26 | 32 | 32 | 32 | 48 | 39 | ... | 54.0 | 44.0 |
| **8324** | 67 | 67 | 1541 | 14 | 28 | 50 | 50 | 50 | 44 | 47 | ... | 53.0 | 72.0 |
| **932** | 77 | 81 | 2085 | 2 | 34 | 64 | 64 | 64 | 76 | 73 | ... | 78.0 | 66.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | .. |
| **11532** | 64 | 64 | 1005 | 11 | 32 | 93 | 93 | 93 | 105 | 102 | ... | 24.0 | 24.0 |
| **16113** | 58 | 68 | 1502 | 11 | 20 | 21 | 21 | 21 | 32 | 26 | ... | 44.0 | 39.0 |
| **14549** | 61 | 73 | 1422 | 11 | 23 | 13 | 13 | 13 | 18 | 14 | ... | 38.0 | 42.0 |
| **14603** | 61 | 84 | 1613 | 9 | 17 | 26 | 26 | 26 | 46 | 39 | ... | 66.0 | 49.0 |
| **11590** | 64 | 64 | 1763 | 1 | 19 | 36 | 36 | 36 | 50 | 43 | ... | 60.0 | 56.0 |

12711 rows × 64 columns

In [35]:

```
X_train.corr()
```

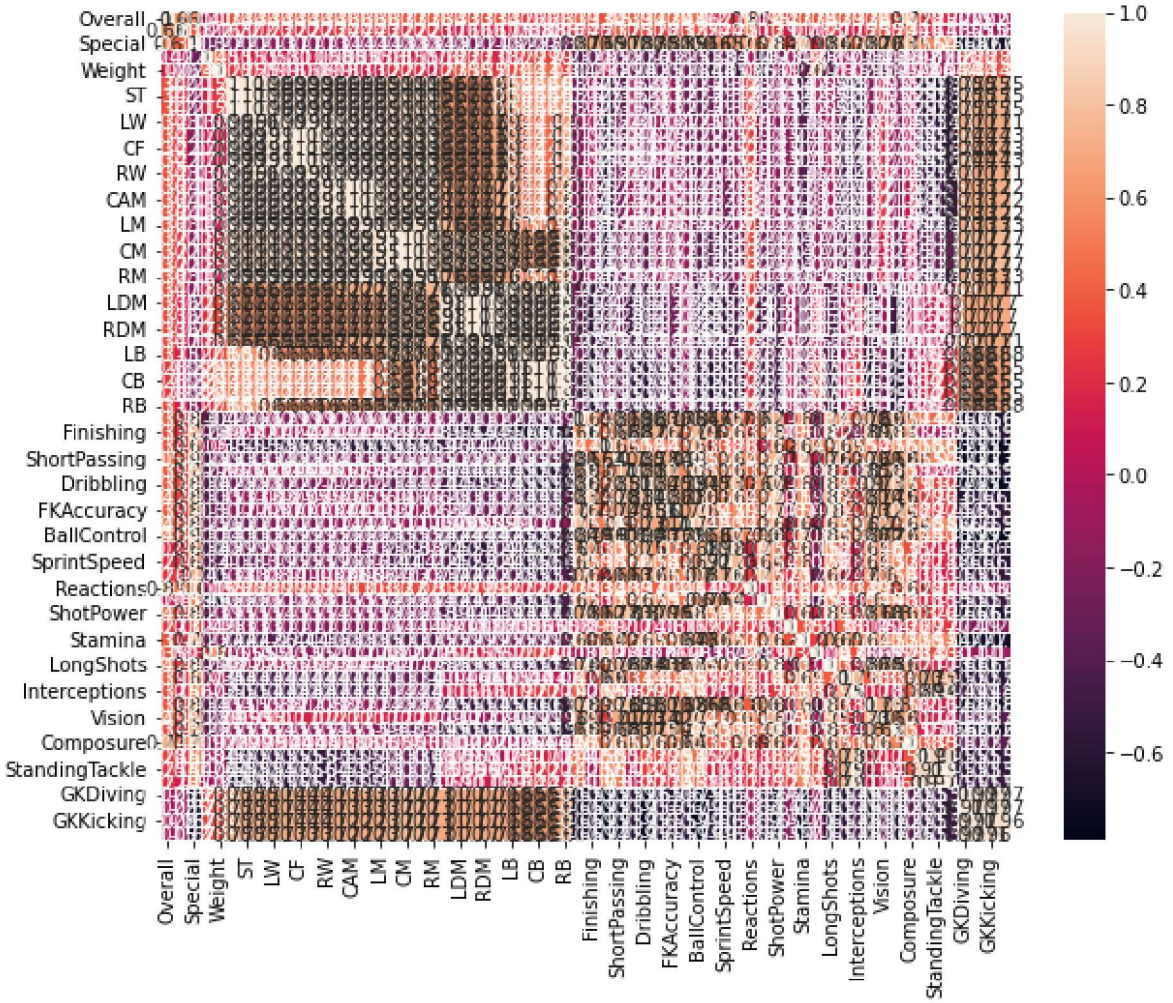
Out[35]:

| | Overall | Potential | Special | Height | Weight | LS | ST | |
|---|---|---|---|---|---|---|---|---|
| **Overall** | 1.000000 | 0.663264 | 0.604935 | 0.040550 | 0.165256 | 0.338139 | 0.338139 | 0.33 |
| **Potential** | 0.663264 | 1.000000 | 0.384688 | 0.021916 | 0.006314 | 0.197538 | 0.197538 | 0.19 |
| **Special** | 0.604935 | 0.384688 | 1.000000 | -0.297463 | -0.265904 | -0.212012 | -0.212012 | -0.21 |
| **Height** | 0.040550 | 0.021916 | -0.297463 | 1.000000 | 0.445577 | 0.154700 | 0.154700 | 0.15 |
| **Weight** | 0.165256 | 0.006314 | -0.265904 | 0.445577 | 1.000000 | 0.250837 | 0.250837 | 0.25 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **SlidingTackle** | 0.217372 | 0.125848 | 0.504970 | -0.080438 | -0.054441 | -0.583721 | -0.583721 | -0.58 |
| **GKDiving** | -0.024528 | -0.054595 | -0.676385 | 0.280936 | 0.344206 | 0.753361 | 0.753361 | 0.75 |
| **GKHandling** | -0.022157 | -0.054619 | -0.674273 | 0.281478 | 0.343832 | 0.753712 | 0.753712 | 0.75 |
| **GKKicking** | -0.028423 | -0.060127 | -0.672329 | 0.277039 | 0.343493 | 0.752924 | 0.752924 | 0.75 |
| **GKPositioning** | -0.014993 | -0.052443 | -0.669847 | 0.280326 | 0.346503 | 0.750599 | 0.750599 | 0.75 |

64 rows × 64 columns

In [36]:

```
import seaborn as sns
#Using Pearson Correlation
plt.figure(figsize=(10,8))
cor = X_train.corr()
sns.heatmap(cor, annot=True)
plt.show()
```



In [37]:

```
# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feature

def correlation(dataset, threshold):#X_train,0.6
    col_corr = set()  # Set of all the names of correlated columns
    col_corr_lst = []
    print(f"set initial {col_corr}")
    print(f"list initial {col_corr_lst}")
    corr_arr = dataset.corr() #corr_arr is my correlaion matrix which is 2d
    for row in range(len(corr_arr)):
        for col in range(row):
            if abs(corr_arr.iloc[row, col]) > threshold: # we are interested in absolute co
                colname = corr_arr.columns[row]  # getting the name of column
                col_corr_lst.append(colname)
                col_corr.add(colname)
                print(f"colname name which is correlated is {colname}")
                print(f"set {col_corr}")
                print(f"lst {col_corr_lst}")


    print(f"list is {col_corr_lst}")
    return col_corr
```

In [38]:

```
corr_features = correlation(X_train, 0.6)#data,threshold
len(set(corr_features))
```

```
ng', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKK
icking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking',
 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicki
ng', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKK
icking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking',
 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicking', 'GKKicki
ng', 'GKKicking', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPo
sitioning', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPosition
ing', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPositioning',
 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPo

sitioning', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPosition
ing', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPositioning',
 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPo
sitioning', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPosition
ing', 'GKPositioning', 'GKPositioning', 'GKPositioning', 'GKPositioning',
 'GKPositioning', 'GKPositioning', 'GKPositioning']
```

Out[38]:

59

In [39]:

```
corr_features
```

Out[39]:

```
{'Acceleration',
 'Aggression',
 'Agility',
 'Balance',
 'BallControl',
 'CAM',
 'CB',
 'CDM',
 'CF',
 'CM',
 'Composure',
 'Crossing',
 'Curve',
 'Dribbling',
 'FKAccuracy',
 'Finishing',
 'GKDiving',
 'GKHandling',
 'GKKicking',
 'GKPositioning',
 'HeadingAccuracy',
 'Interceptions',
 'LAM',
 'LB',
 'LCB',
 'LCM',
 'LDM',
 'LF',
 'LM',
 'LW',
 'LWB',
 'LongPassing',
 'LongShots',
 'Marking',
 'Penalties',
 'Positioning',
 'Potential',
 'RAM',
 'RB',
 'RCB',
 'RCM',
 'RDM',
 'RF',
 'RM',
 'RS',
 'RW',
 'RWB',
 'Reactions',
 'ST',
 'ShortPassing',
 'ShotPower',
 'SlidingTackle',
 'Special',
 'SprintSpeed',
 'Stamina',
```

```
 'StandingTackle',
 'Strength',
 'Vision',
 'Volleys'}
```

In [40]:

```
X_train.drop(corr_features,axis=1,inplace = True)
X_test.drop(corr_features,axis=1,inplace = True)
```

In [41]:

```
X_train
```

Out[41]:

| | Overall | Height | Weight | LS | Jumping |
|---|---|---|---|---|---|
| **12383** | 63 | 2 | 23 | 34 | 69.0 |
| **8471** | 67 | 13 | 35 | 28 | 76.0 |
| **7381** | 68 | 11 | 26 | 32 | 75.0 |
| **8324** | 67 | 14 | 28 | 50 | 82.0 |
| **932** | 77 | 2 | 34 | 64 | 68.0 |
| **...** | ... | ... | ... | ... | ... |
| **11532** | 64 | 11 | 32 | 93 | 58.0 |
| **16113** | 58 | 11 | 20 | 21 | 65.0 |
| **14549** | 61 | 11 | 23 | 13 | 84.0 |
| **14603** | 61 | 9 | 17 | 26 | 60.0 |
| **11590** | 64 | 1 | 19 | 36 | 59.0 |

12711 rows × 5 columns

In [42]:

```
X_test
```

Out[42]:

| | Overall | Height | Weight | LS | Jumping |
|---|---|---|---|---|---|
| **12454** | 63 | 1 | 17 | 32 | 50.0 |
| **10090** | 65 | 9 | 19 | 36 | 63.0 |
| **15674** | 59 | 8 | 18 | 34 | 56.0 |
| **14242** | 61 | 11 | 28 | 30 | 78.0 |
| **2009** | 75 | 1 | 22 | 50 | 70.0 |
| **...** | ... | ... | ... | ... | ... |
| **2133** | 74 | 2 | 18 | 50 | 72.0 |
| **799** | 78 | 1 | 19 | 52 | 66.0 |
| **9474** | 66 | 9 | 16 | 48 | 89.0 |
| **5484** | 70 | 2 | 28 | 44 | 73.0 |
| **14572** | 61 | 11 | 31 | 9 | 52.0 |

5448 rows × 5 columns

# Apply the variance and remove the columns whose variance is less that 20%

In [43]:

```
### It will zero variance features
from sklearn.feature_selection import VarianceThreshold
var_thres=VarianceThreshold(threshold=0.2)
var_thres.fit(X_train)
```

Out[43]:

```
VarianceThreshold(threshold=0.2)
```

In [44]:

```
#which column is having good variaty of data means good variance
var_thres.get_support()
```

Out[44]:

```
array([ True,  True,  True,  True,  True])
```

In [45]:

```
X_train.columns[var_thres.get_support() == True]
```

Out[45]:

```
Index(['Overall', 'Height', 'Weight', 'LS', 'Jumping'], dtype='object')
```

In [46]:

```
columns_having_var_less_than_20 = X_train.columns[var_thres.get_support() == True]
```

In [47]:

```
columns_having_var_less_than_20
```

Out[47]:

```
Index(['Overall', 'Height', 'Weight', 'LS', 'Jumping'], dtype='object')
```

In [48]:

```
X_train.columns[var_thres.get_support() == False] #code to get constant columns
```

Out[48]:

```
Index([], dtype='object')
```

In [49]:

```
constant_columns = X_train.columns[var_thres.get_support() == False]
```

In [50]:

```
constant_columns
```

Out[50]:

```
Index([], dtype='object')
```

In [51]:

```
len(columns_having_var_less_than_20)
```

Out[51]:

```
5
```

In [52]:

```
### It will zero variance features
from sklearn.feature_selection import VarianceThreshold
var_thres=VarianceThreshold(threshold=0.2)
var_thres.fit(X_test)
```

Out[52]:

```
VarianceThreshold(threshold=0.2)
```

In [53]:

```
#which column is having good variaty of data means good variance
var_thres.get_support()
```

Out[53]:

```
array([ True,  True,  True,  True,  True])
```

In [54]:

```
X_test.columns[var_thres.get_support() == True]
```

Out[54]:

```
Index(['Overall', 'Height', 'Weight', 'LS', 'Jumping'], dtype='object')
```

In [55]:

```
columns_having_var_less_than_20 = X_test.columns[var_thres.get_support() == True]
```

In [56]:

```
columns_having_var_less_than_20
```

Out[56]:

```
Index(['Overall', 'Height', 'Weight', 'LS', 'Jumping'], dtype='object')
```

In [57]:

```
X_test.columns[var_thres.get_support() == False] #code to get constant columns
```

Out[57]:

```
Index([], dtype='object')
```

In [58]:

```
constant_columns = X_test.columns[var_thres.get_support() == False]
```

In [59]:

```
constant_columns
```

Out[59]:

```
Index([], dtype='object')
```

In [60]:

```
len(columns_having_var_less_than_20)
```

Out[60]:

```
5
```

# Conclusion : The dataset is correlated in that remove the 59 correlated columns and in the variance method none of the column is removed.