In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
loan_data=pd.read_csv(r"C:\Users\Pranav\Desktop\Prachi\titanic-dataset\loan.csv")
appln_data=pd.read_csv(r"C:\Users\Pranav\Desktop\Prachi\titanic-dataset\applicant.csv")
```

In [3]:

```
loan_data.columns # this shows the column names
```

Out[3]:

```
Index(['loan_application_id', 'applicant_id', 'Months_loan_taken_for',
       'Purpose', 'Principal_loan_amount',
       'EMI_rate_in_percentage_of_disposable_income', 'Property',
       'Has_coapplicant', 'Has_guarantor', 'Other_EMI_plans',
       'Number_of_existing_loans_at_this_bank', 'Loan_history',
       'high_risk_applicant'],
      dtype='object')
```

In [5]:

```
appln_data.columns
```

Out[5]:

```
Index(['applicant_id', 'Primary_applicant_age_in_years', 'Gender',
       'Marital_status', 'Number_of_dependents', 'Housing',
       'Years_at_current_residence', 'Employment_status',
       'Has_been_employed_for_at_least', 'Has_been_employed_for_at_most',
       'Telephone', 'Foreign_worker', 'Savings_account_balance',
       'Balance_in_existing_bank_account_(lower_limit_of_bucket)',
       'Balance_in_existing_bank_account_(upper_limit_of_bucket)'],
      dtype='object')
```

In [6]:

```
main=pd.merge(loan_data,appln_data,on="applicant_id")  #merging two data sets by common column
```

In [7]:

```
main.head()
```

Out[7]:

	loan_application_id	applicant_id	Months_loan_taken_for	Purpose	Principal_loan_amount	EMI_rate_in_percentage_of_disposable_income	
0	d68d975e-edad-11ea-8761-1d6f9c1ff461	1469590	6	electronic equipment	1169000	4	
1	d68d989e-edad-11ea-b1d5-2bcf65006448	1203873	48	electronic equipment	5951000	2	
2	d68d995c-edad-11ea-814a-1b6716782575	1432761	12	education	2096000	2	
3	d68d99fc-edad-11ea-8841-17e8848060ae	1207582	42	FF&E	7882000	2	agr
4	d68d9a92-edad-11ea-9f3d-1f8682db006a	1674436	24	new vehicle	4870000	3	

5 rows × 27 columns

In [8]:

```
rows=main.shape[0]
r_null=main.isnull().sum()
r_null/rows*100
```

Out[8]:

```
loan_application_id                                   0.0
applicant_id                                          0.0
Months_loan_taken_for                                 0.0
Purpose                                               1.2
Principal_loan_amount                                 0.0
EMI_rate_in_percentage_of_disposable_income           0.0
Property                                             15.4
Has_coapplicant                                       0.0
Has_guarantor                                         0.0
Other_EMI_plans                                      81.4
Number_of_existing_loans_at_this_bank                 0.0
Loan_history                                          0.0
high_risk_applicant                                   0.0
Primary_applicant_age_in_years                        0.0
Gender                                                0.0
Marital_status                                        0.0
Number_of_dependents                                  0.0
Housing                                               0.0
Years_at_current_residence                            0.0
Employment_status                                     0.0
Has_been_employed_for_at_least                        6.2
Has_been_employed_for_at_most                        25.3
Telephone                                            59.6
Foreign_worker                                        0.0
Savings_account_balance                              18.3
Balance_in_existing_bank_account_(lower_limit_of_bucket)    66.8
Balance_in_existing_bank_account_(upper_limit_of_bucket)    45.7
dtype: float64
```

In [9]:

```
main=main.drop(columns=["Other_EMI_plans","Has_been_employed_for_at_most","Telephone","Balance_in_existing_bank_account_(lower_li
#droping the columns whose null values are above 25%
```

In [10]:

```
main.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                                       Non-Null Count  Dtype 
---  ------                                       --------------  ----- 
 0   loan_application_id                          1000 non-null   object
 1   applicant_id                                 1000 non-null   int64 
 2   Months_loan_taken_for                        1000 non-null   int64 
 3   Purpose                                      988 non-null    object
 4   Principal_loan_amount                        1000 non-null   int64 
 5   EMI_rate_in_percentage_of_disposable_income  1000 non-null   int64 
 6   Property                                     846 non-null    object
 7   Has_coapplicant                              1000 non-null   int64 
 8   Has_guarantor                                1000 non-null   int64 
 9   Number_of_existing_loans_at_this_bank        1000 non-null   int64 
 10  Loan_history                                 1000 non-null   object
 11  high_risk_applicant                          1000 non-null   int64 
 12  Primary_applicant_age_in_years               1000 non-null   int64 
 13  Gender                                       1000 non-null   object
 14  Marital_status                               1000 non-null   object
 15  Number_of_dependents                         1000 non-null   int64 
 16  Housing                                      1000 non-null   object
 17  Years_at_current_residence                   1000 non-null   int64 
 18  Employment_status                            1000 non-null   object
 19  Has_been_employed_for_at_least               938 non-null    object
 20  Foreign_worker                               1000 non-null   int64 
 21  Savings_account_balance                      817 non-null    object
dtypes: int64(12), object(10)
memory usage: 179.7+ KB
```

In [12]:

```
main["Purpose"]=main["Purpose"].fillna(main["Purpose"].mode()[0])
main["Property"]=main["Property"].fillna(main["Property"].mode()[0])
main["Has_been_employed_for_at_least"]=main["Has_been_employed_for_at_least"].fillna(main["Has_been_employed_for_at_least"].mode(
main["Savings_account_balance"]=main["Savings_account_balance"].fillna(main["Savings_account_balance"].mode()[0])
```

In [13]:

```
main.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column                                     Non-Null Count  Dtype 
---  ------                                     --------------  ----- 
 0   loan_application_id                        1000 non-null   object
 1   applicant_id                               1000 non-null   int64 
 2   Months_loan_taken_for                      1000 non-null   int64 
 3   Purpose                                    1000 non-null   object
 4   Principal_loan_amount                      1000 non-null   int64 
 5   EMI_rate_in_percentage_of_disposable_income  1000 non-null   int64 
 6   Property                                   1000 non-null   object
 7   Has_coapplicant                            1000 non-null   int64 
 8   Has_guarantor                              1000 non-null   int64 
 9   Number_of_existing_loans_at_this_bank      1000 non-null   int64 
 10  Loan_history                               1000 non-null   object
 11  high_risk_applicant                        1000 non-null   int64 
 12  Primary_applicant_age_in_years             1000 non-null   int64 
 13  Gender                                     1000 non-null   object
 14  Marital_status                             1000 non-null   object
 15  Number_of_dependents                       1000 non-null   int64 
 16  Housing                                    1000 non-null   object
 17  Years_at_current_residence                 1000 non-null   int64 
 18  Employment_status                          1000 non-null   object
 19  Has_been_employed_for_at_least             1000 non-null   object
 20  Foreign_worker                             1000 non-null   int64 
 21  Savings_account_balance                    1000 non-null   object
dtypes: int64(12), object(10)
memory usage: 179.7+ KB
```

In [14]:

```
#Data preprocessing
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
object_list=main.select_dtypes(include=['object']).columns
for i in object_list:
    main[i]=le.fit_transform(main[i])
```

In [15]:

```
#slicing the target column, target column is y and rest is x
y=main["high_risk_applicant"]
x=main.drop(columns=["high_risk_applicant"],axis=1)
```

In [16]:

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

In [17]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [18]:

```
#preparing model to check the dependency of survival on all the columns
from sklearn.linear_model import Ridge, Lasso, LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor

list_algo = [LinearRegression(), Ridge(), Lasso(), KNeighborsRegressor(), DecisionTreeRegressor()]

for algo in list_algo:
    model=algo
    model.fit(x_train, y_train)
    y_pred=model.predict(x_test)
    print(f'The score of {algo} is {algo.score(x_test,y_test)*100}%')
```

```
The score of LinearRegression() is 2.6671258251008023%
The score of Ridge() is 2.689354223527529%
The score of Lasso() is -0.17742371888986863%
The score of KNeighborsRegressor() is -8.824168658987208%
The score of DecisionTreeRegressor() is -67.89828298196045%
```

In [ ]:

```

```