Predicting Personal Loan Approval Using Machine Learning

1.INTRODUCTION

1.1 Overview

*A loan is a sum of money that is borrowed and repaid over a period of time, typically with interest. There are various types of loans available to individuals and businesses, such as personal loans, mortgages, auto loans, student loans, business loans and many more.They are offered by banks, credit unions, and other financial institutions, and the terms of the loan, such as interest rate, repayment period, and fees, vary depending on the lender and the type of loan.

*Home repairs, medical expenses, debt consolidation, and more. The loan amount, interest rate, and repayment period vary depending on the lender and the borrower's creditworthiness.To qualify for a personal loan, borrowers typically need to provide proof of income and have a good credit score.

*Predicting personal loan approval using machine learning analyses a borrower's financial data and credit history to determine the likelihood of loan approval. This can help financial institutions to make more informed decisions about which loan applications to approve and which to deny.

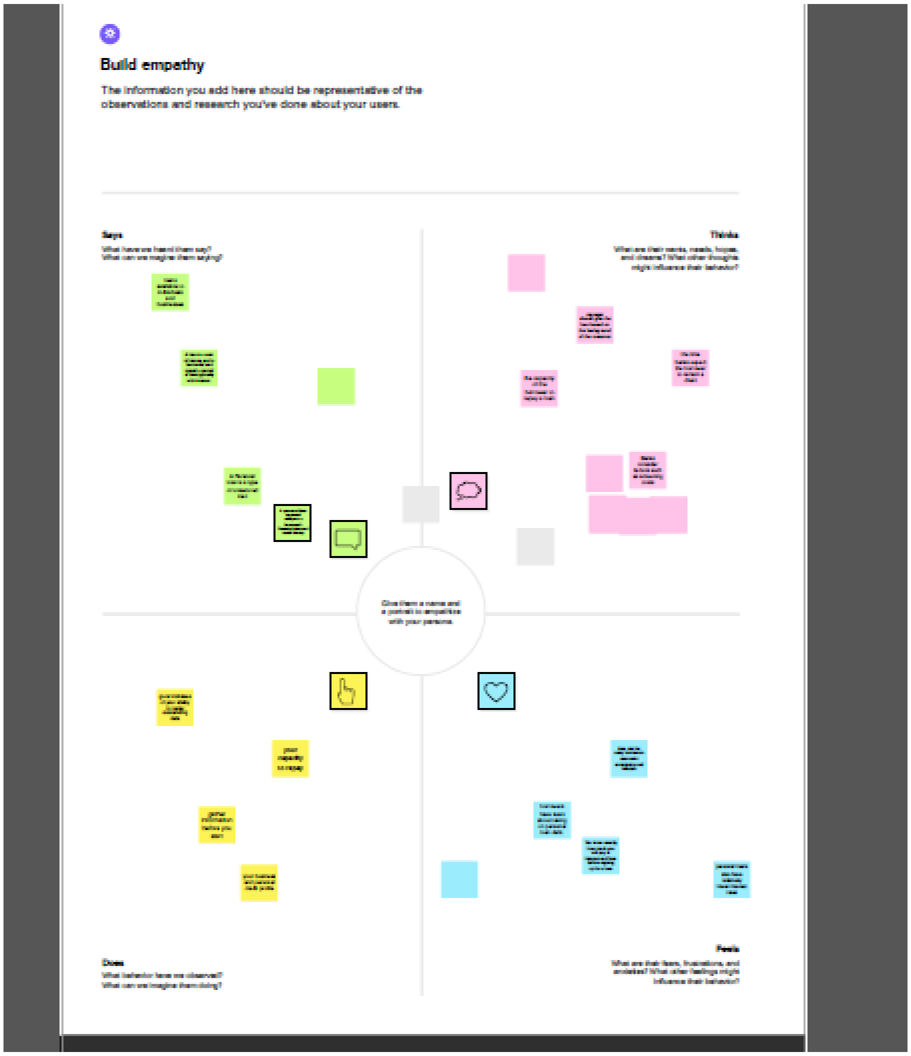
1.b)PURPOSE

1. Data consolidation

2. Emergency expenses

2.Problem Definition & Design Thinking

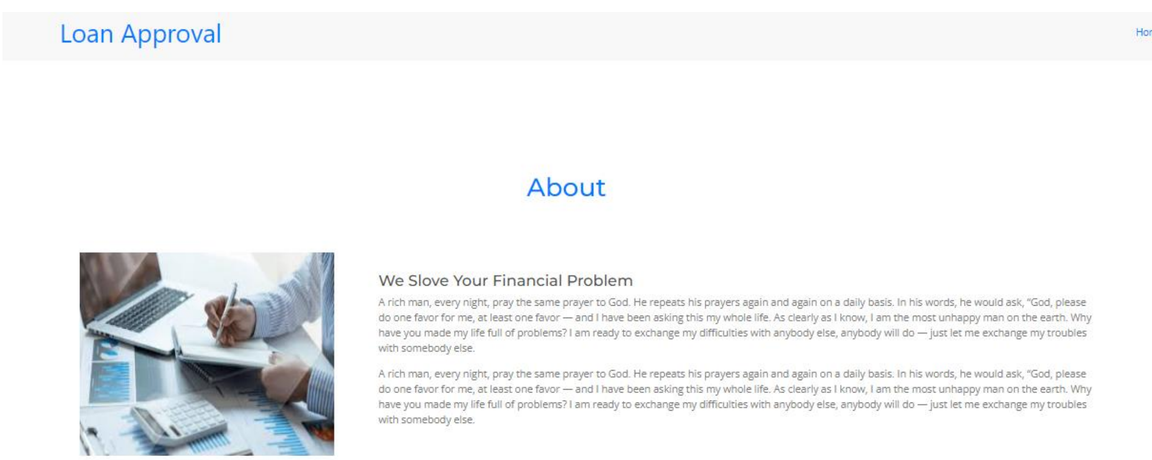
2.1 Empathy Map



2.2 Ideation & Brainstorming


PROBLEM
1.How we might find the personalnloan approval?
PROBLEM
2.How to get an easy personal loan?
check your credit score
PROBLEM
3.How to solve loan problems?
identify the problem
lower your expenses
pay in cash
avoid buying new
PROBLEM
4.What are the a personal loan approval involes the various steps?
loan application
Discussion with the lender
Background verification by the bank
PROBLEM
5.who is eligible for pre_approved loan?
high credit score
a good credit history
stable income
excellent repayment record

3.RESULT


Loan Approval
Hon
About
We Slove Your Financial Problem
A rich man, every night, pray the same prayer to God. He repeats his prayers again and again on a daily basis. In his words, he would ask, "God, please do one favor for me, at least one favor — and I have been asking this my whole life. As clearly as I know, I am the most unhappy man on the earth. Why have you made my life full of problems? I am ready to exchange my difficulties with anybody else, anybody will do — just let me exchange my troubles with somebody else.
A rich man, every night, pray the same prayer to God. He repeats his prayers again and again on a daily basis. In his words, he would ask, "God, please do one favor for me, at least one favor — and I have been asking this my whole life. As clearly as I know, I am the most unhappy man on the earth. Why have you made my life full of problems? I am ready to exchange my difficulties with anybody else, anybody will do — just let me exchange my troubles with somebody else.

Gallery












Loan Approval How it works?

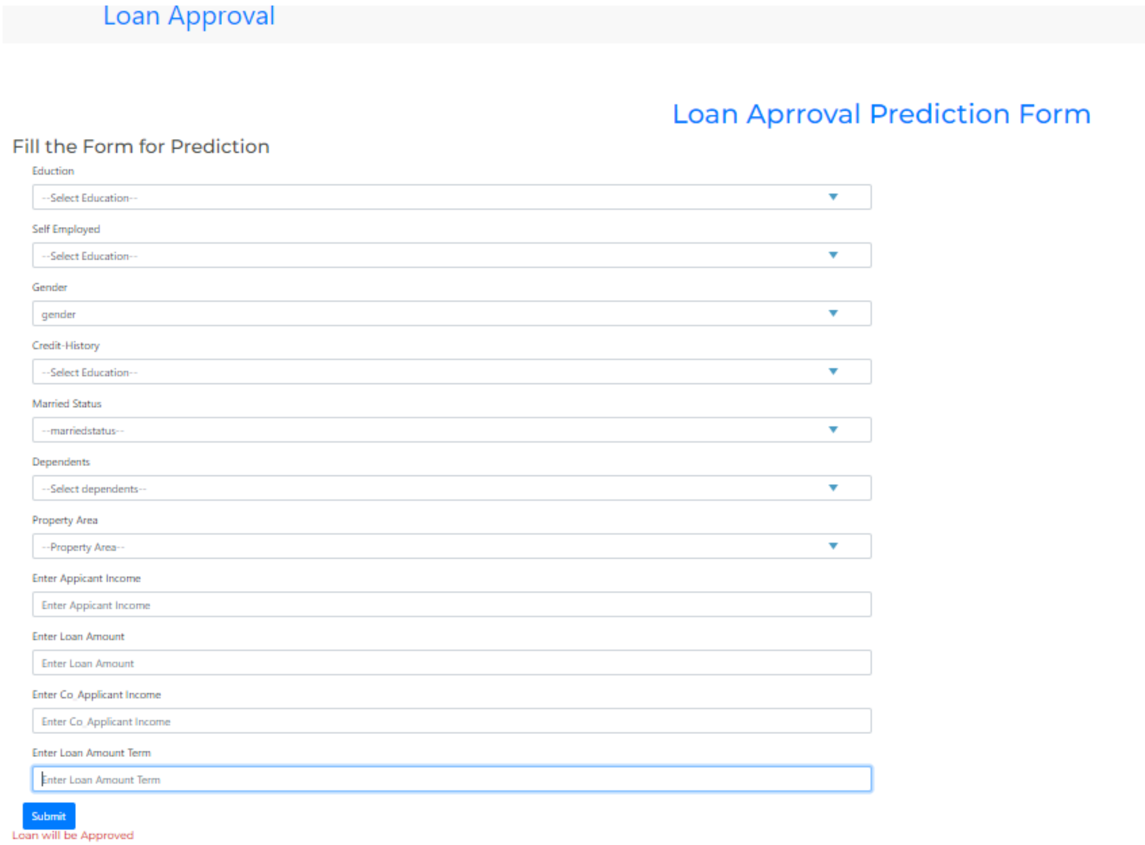
A rich man, every night, pray the same prayer to God. He repeats his prayers again and again on a daily basis. In his words, he would ask, "God, please do one favor for me, at least one favor — and I have been asking this my whole life. As clearly as I know, I am the most unhappy man on the earth. Why have you made my life full of problems? I am ready to exchange my difficulties with anybody else, anybody will do — just let me exchange my troubles with somebody else.

Contact US

No.3, Big street, chinna kukundi village, arcot taluk, vellore, Tamil nadu -632503

+91 8147786347

pthulasi10@gmail.com



Loan Approval

Loan Aprroval Prediction Form

Fill the Form for Prediction

Eduction

--Select Education--

Self Employed

--Select Education--

Gender

gender

Credit-History

--Select Education--

Married Status

--marriedstatus--

Dependents

--Select dependents--

Property Area

--Property Area--

Enter Appicant Income

Enter Appicant Income

Enter Loan Amount

Enter Loan Amount

Enter Co_Applicant Income

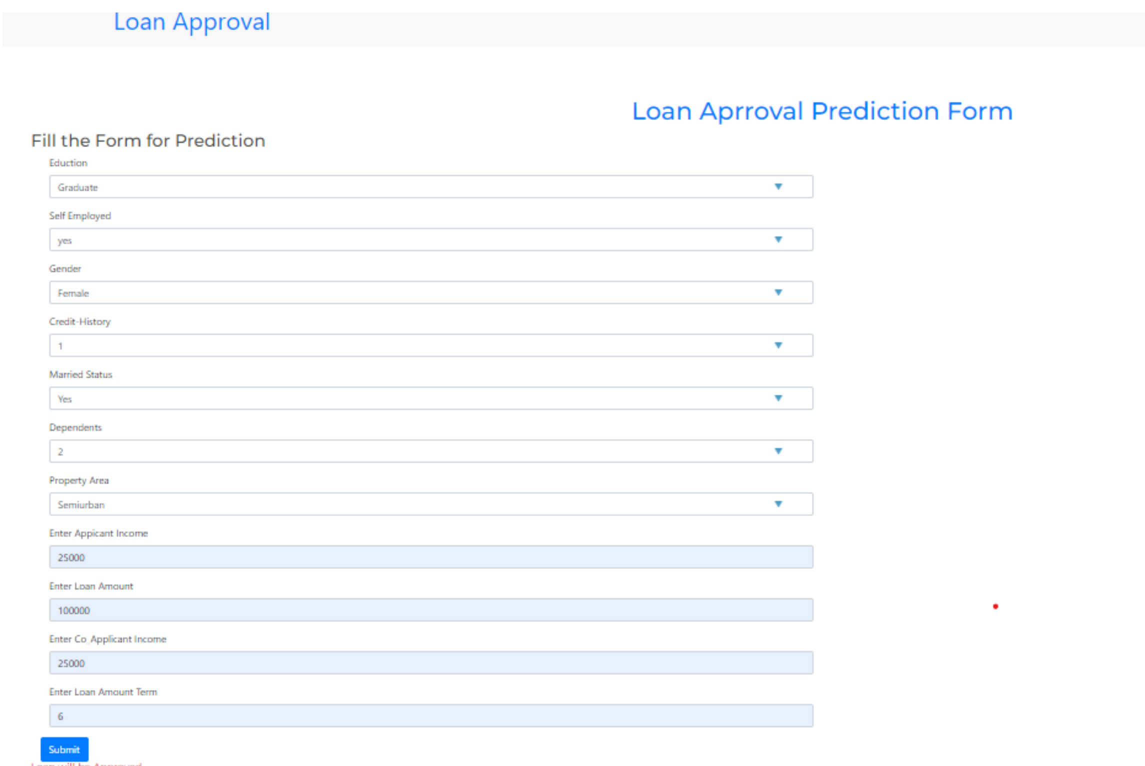
Enter Co_Applicant Income

Enter Loan Amount Term

Enter Loan Amount Term

Submit

Loan will be Approved



Loan Approval

Loan Aprroval Prediction Form

Fill the Form for Prediction

Eduction

Graduate

Self Employed

yes

Gender

Female

Credit-History

1

Married Status

Yes

Dependents

2

Property Area

Semiurban

Enter Appicant Income

25000

Enter Loan Amount

100000

Enter Co_Applicant Income

25000

Enter Loan Amount Term

6

Submit

Loan will be Approved

4. ADVANTAGES AND DISADVANTAGES OF LOAN APPROVAL

ADVANTAGE

* Spread the cost of a significant purchase safely.
* Can help you manage your personal finances.
* Ideal if you have struggled to save in the past.
* Unsecured loans are not tied to assets

DISADVANTAGE

* Loan term commitment.
* Good product requires a good credit score.
* Certain loan types are risker than others.
* Will never get 0% interest unlike a credit card or finance deal.

5. APPLICATIONS

1.Finnable

2.NIRA instant Personal Loan App

3.mPokket

4.Fullerton india InstaLoan

6.conclusion

*From the proper view of analysis this system can be used can be perfect for detection of clients who are eligiable for approval of loan.the software is working perfect and can be used for all banking requirements.Thissystem can be easily uploaded in any operating system.

* Since the technology is moving towards online,this system has more scope for the upcoming days.This system is more secure and reliable.since we have used random forest algorithm the system returns accurate results.

7.FUTURE SCOPE

*Today fast growing IT sector requires the development of new technology and the updating of existing technology that allows us to eliminate human interference and boost job productivity.

*This model is used for the banking system or anyone who wants to apply for a loan.Based on the examination of the data,it is apparentr that is reduces all frauds committed during the loan approval.

*Time is valuable to everyone ,and by doing so,not only the bank,but also applicants time will be reduced.

8.Appendix

A.Source code & Output

importing necessary libraries

```
import pandas as pd
import numpy as py
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier,RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix, f1_score
```

Importing the dataset

```
data = pd.read_csv('/content/train_u6lujuX_CVtuZ9i.csv')
data
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Am
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	
...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0.0	Graduate	No	2900	0.0	71.0	
610	LP002979	Male	Yes	3.0	Graduate	No	4106	0.0	40.0	
611	LP002983	Male	Yes	1.0	Graduate	No	8072	240.0	253.0	
612	LP002984	Male	Yes	2.0	Graduate	No	7583	0.0	187.0	
613	LP002990	Female	No	0.0	Graduate	Yes	4583	0.0	133.0	

614 rows × 13 columns

```
data.drop(['Loan_ID'],axis=1,inplace=True)
```

```
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0
1	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0
2	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0
3	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0
4	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0

```
data['Gender']=data['Gender'].map({'Female':1,'Male':0})
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.0	No	0.0	Graduate	No	5849	0.0	NaN	360.0
1	0.0	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0
2	0.0	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0
3	0.0	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0
4	0.0	No	0.0	Graduate	No	6000	0.0	141.0	360.0

```
data['Property_Area']=data['Property_Area'].map({'Urban':2,'Semiurban':1,'Rural':0})
data.head()
```

```python
data['Property_Area']=data['Property_Area'].map({'Urban':2,'Semiurban':1,'Rural':0})
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.0	No	0.0	Graduate	No	5849	0.0	NaN	360.0
1	0.0	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0
2	0.0	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0
3	0.0	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0
4	0.0	No	0.0	Graduate	No	6000	0.0	141.0	360.0

```python
data['Married']=data['Married'].map({'Yes':1,'No':0})
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.0	0.0	0.0	Graduate	No	5849	0.0	NaN	360.0
1	0.0	1.0	1.0	Graduate	No	4583	1508.0	128.0	360.0
2	0.0	1.0	0.0	Graduate	Yes	3000	0.0	66.0	360.0
3	0.0	1.0	0.0	Not Graduate	No	2583	2358.0	120.0	360.0
4	0.0	0.0	0.0	Graduate	No	6000	0.0	141.0	360.0

```python
data['Education']=data['Education'].map({'Graduate':1,'Not Graduate':0})
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.0	0.0	0.0	1	No	5849	0.0	NaN	360.0
1	0.0	1.0	1.0	1	No	4583	1508.0	128.0	360.0
2	0.0	1.0	0.0	1	Yes	3000	0.0	66.0	360.0
3	0.0	1.0	0.0	0	No	2583	2358.0	120.0	360.0
4	0.0	0.0	0.0	1	No	6000	0.0	141.0	360.0

```python
data['Self_Employed']=data['Self_Employed'].map({'Yes':1,'No':0})
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.0	0.0	0.0	1	0.0	5849	0.0	NaN	360.0
1	0.0	1.0	1.0	1	0.0	4583	1508.0	128.0	360.0
2	0.0	1.0	0.0	1	1.0	3000	0.0	66.0	360.0
3	0.0	1.0	0.0	0	0.0	2583	2358.0	120.0	360.0
4	0.0	0.0	0.0	1	0.0	6000	0.0	141.0	360.0

```python
data['Loan_Status']=data['Loan_Status'].map({'Y':1,'N':0})
data.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	0.0	0.0	0.0	1	0.0	5849	0.0	NaN	360.0
1	0.0	1.0	1.0	1	0.0	4583	1508.0	128.0	360.0
2	0.0	1.0	0.0	1	1.0	3000	0.0	66.0	360.0
3	0.0	1.0	0.0	0	0.0	2583	2358.0	120.0	360.0
4	0.0	0.0	0.0	1	0.0	6000	0.0	141.0	360.0

```python
data.isnull().sum()
```

```
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
```

```python
data['Gender'] = data['Gender'].fillna(data['Gender'].mode()[0])
```

```python
data['Married']=data['Married'].fillna(data['Married'].mode()[0])
data['Dependents']=data['Dependents'].fillna(data['Dependents'].mode()[0])
data['Self_Employed']=data['Self_Employed'].fillna(data['Self_Employed'].mode()[0])
data['LoanAmount']=data['LoanAmount'].fillna(data['LoanAmount'].mode()[0])
data['Loan_Amount_Term']=data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mode()[0])
data['Credit_History']=data['Credit_History'].fillna(data['Credit_History'].mode()[0])
```

```
data.isnull().sum()
```

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             614 non-null    float64
 1   Married            614 non-null    float64
 2   Dependents         614 non-null    float64
 3   Education          614 non-null    int64  
 4   Self_Employed      614 non-null    float64
 5   ApplicantIncome    614 non-null    int64  
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         614 non-null    float64
 8   Loan_Amount_Term   614 non-null    float64
 9   Credit_History     614 non-null    float64
 10  Property_Area      614 non-null    int64  
 11  Loan_Status        614 non-null    int64  
dtypes: float64(8), int64(4)
memory usage: 57.7 KB
```

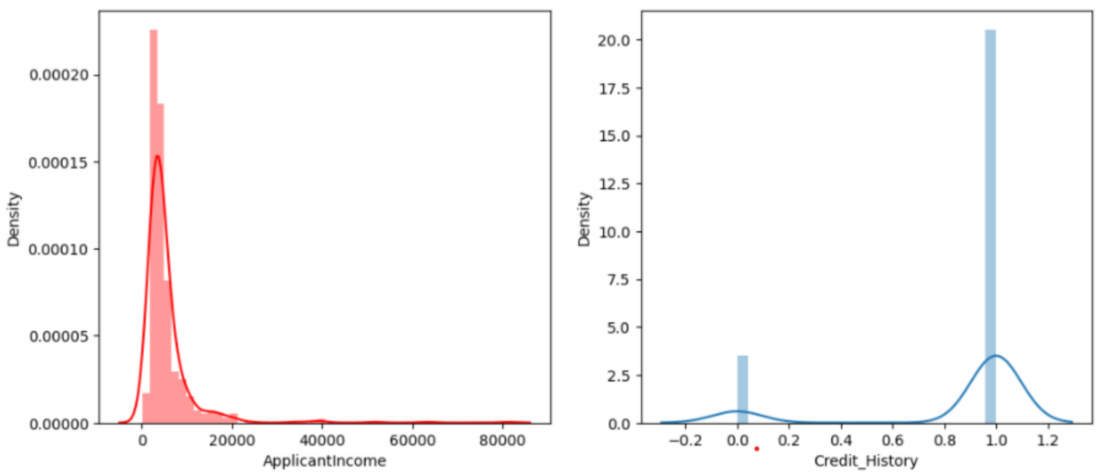
```
data['ApplicantIncome']=data['ApplicantIncome'].astype('float64')
```

```
data['Gender']=data['Gender'].astype('object')
```

```
data['Married']=data['Married'].astype('object')
data['CoapllicantIncome']=data['CoapplicantIncome'].astype('object')
```

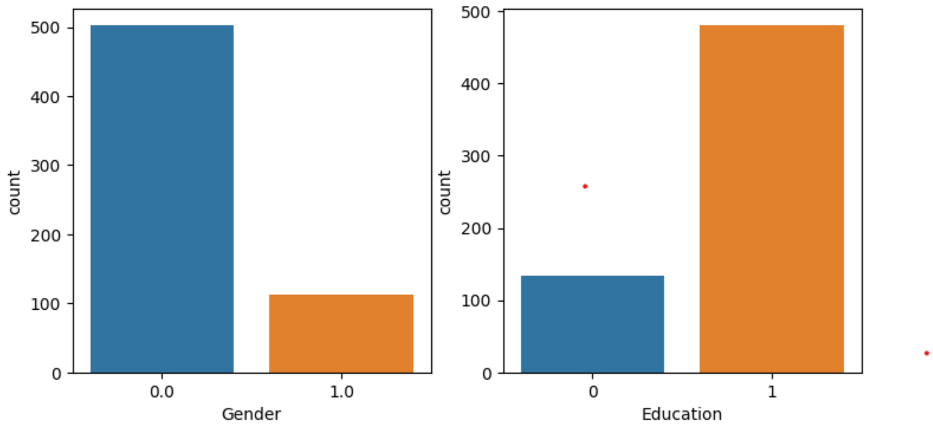
```
plt.figure(figsize=(12,5))
plt.subplot(121)
sns.distplot(data['ApplicantIncome'],color='r')
plt.subplot(122)
sns.distplot(data['Credit_History'])
plt.show()
```

```
<ipython-input-19-28d1357886bf>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['ApplicantIncome'],color='r')
<ipython-input-19-28d1357886bf>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['Credit_History'])
```



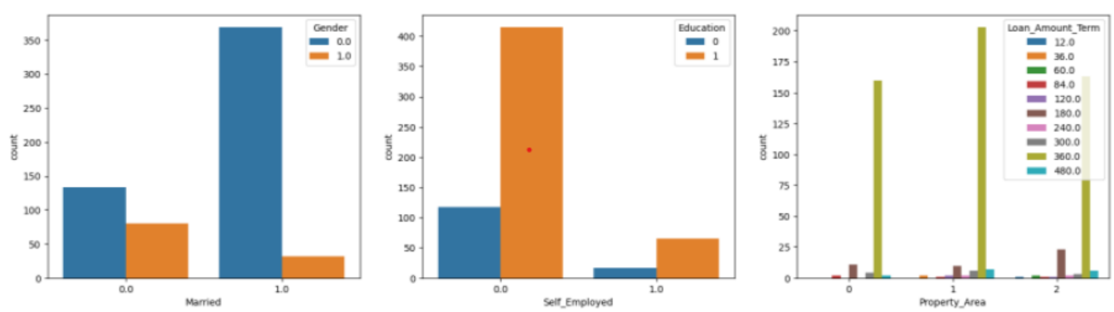
ApplicantIncome Credit_History

In [20]:
```python
plt.figure(figsize=(18,4))
plt.subplot(1,4,1)
sns.countplot(x='Gender',data=data)
plt.subplot(1,4,2)
sns.countplot(x='Education',data=data)
plt.show()
```




```python
plt.figure(figsize=(20,5))
plt.subplot(131)
sns.countplot(x='Married',hue='Gender',data=data)
plt.subplot(132)
sns.countplot(x='Self_Employed',hue='Education',data=data)
plt.subplot(133)
sns.countplot(x='Property_Area',hue='Loan_Amount_Term',data=data)
```

```
<Axes: xlabel='Property_Area', ylabel='count'>
```




```python
pd.crosstab(data['Gender'],[data['Self_Employed']])
```

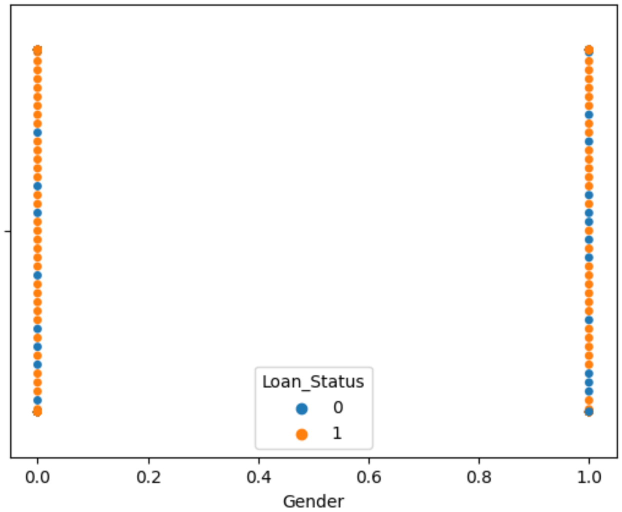
Self_Employed	0.0	1.0
Gender		
0.0	435	67
1.0	97	15

```python
sns.swarmplot(x='Gender',data=data, hue='Loan_Status')
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 21.7% of the points cannot be placed; you ma
y want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

```
<Axes: xlabel='Gender'>
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/categorical.py:3544: UserWarning: 86.6% of the points cannot be placed; you ma
y want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```




```python
from imblearn.combine import SMOTETomek
```

```python
smote=SMOTETomek()
```

```python
y=data['Loan_Status']
x=data.drop(columns=['Loan_Status'],axis=1)
```

```python
x.shape
```

```
(614, 12)
```

```python
y.shape
```

```
(614,)
```

```python
x_bal,y_bal=smote.fit_resample(x,y)
```

```python
print(y.value_counts())
print(y_bal.value_counts())
```

```
1    422
0    192
Name: Loan_Status, dtype: int64
1    353
0    353
Name: Loan_Status, dtype: int64
```

```python
names = x_bal.columns
```

```python
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
```

```
x_bal=pd.DataFrame(x_bal,columns=names)
x_bal.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Tern
0	-0.527179	-1.328670	-0.781675	0.644073	-0.392474	0.128583	-0.474599	-0.271093	0.29398
1	-0.527179	0.808091	0.250997	0.644073	-0.392474	-0.092841	-0.035134	-0.164657	0.29398
2	-0.527179	0.808091	-0.781675	0.644073	2.761483	-0.369708	-0.474599	-0.989536	0.29398
3	-0.527179	0.808091	-0.781675	-1.552620	-0.392474	-0.442641	0.212576	-0.271093	0.29398
4	-0.527179	-1.328670	-0.781675	0.644073	-0.392474	0.154993	-0.474599	0.008302	0.29398

```
X_train,X_test,y_train,y_test=train_test_split(x_bal,y_bal,test_size=0.33,random_state=42)
```

```
X_train.shape
```

```
(473, 12)
```

```
X_test.shape
```

```
(233, 12)
```

```
X_train.shape,y_test.shape
```

```
((473, 12), (233,))
```

```
def RandomForest(X_train,X_test,y_train,y_test):
  model = RandomForestClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))
```

```
RandomForest(X_train,X_test,y_train,y_test)
```

```
..0
).8669527896995708
```

```
def DecisionTree(X_train,X_test,y_train,y_test):
  model = DecisionTreeClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))
```

```
DecisionTree(X_train,X_test,y_train,y_test)
```

```
1.0
0.776824034334764
```

```python
def KNN(X_train,X_test,y_train,y_test):
  model = KNeighborsClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))
```

```python
KNN(X_train,X_test,y_train,y_test)
```

```
).8308668076109936
).7467811158798283
```

```python
def XGB(X_train,X_test,y_train,y_test):
  model = GradientBoostingClassifier()
  model.fit(X_train,y_train)
  y_tr = model.predict(X_train)
  print(accuracy_score(y_tr,y_train))
  yPred = model.predict(X_test)
  print(accuracy_score(yPred,y_test))
```

```python
XGB(X_train,X_test,y_train,y_test)
```

```
0.9408033826638478
0.8068669527896996
```

```python
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.linear_model import LinearRegression
linear_regressor=LinearRegression()
```

```
linear_regressor.fit(X_train,y_train)
linear_regressor.predict(X_test)
```

```
array([ 7.62887717e-01,  4.17309207e-01,  7.19513090e-01,  7.15054853e-01,
        3.79980456e-01,  1.13424505e-01,  8.39451152e-01,  8.46231370e-01,
        2.84978418e-01,  6.41875966e-01,  8.62406332e-02,  5.99742671e-01,
        4.28457746e-01,  7.15706589e-01,  7.29724974e-01,  6.13714419e-01,
        4.60024696e-01,  4.20990567e-01,  8.96734460e-01,  6.46095902e-01,
        8.62049107e-01,  7.13237038e-01,  2.48144635e-01,  7.32651822e-01,
       -1.85434809e-01,  6.73401457e-01,  6.78942563e-01,  6.92901821e-01,
        5.19398839e-01,  2.70146671e-01,  7.36718111e-01,  8.74465191e-01,
        2.76258230e-01,  4.07425153e-01,  3.56785605e-02,  7.18841330e-01,
        6.91429527e-01,  5.90565303e-01,  5.69260942e-01,  5.24007715e-01,
        2.34339507e-01,  4.48992953e-01,  8.77050640e-01, -1.50655047e-01,
        7.51816236e-01,  7.40139858e-01,  5.34155287e-01,  8.28172811e-01,
        2.71768208e-01,  8.39549293e-01,  1.23728894e-02,  4.07476714e-01,
        6.12313206e-01,  5.74243516e-01,  1.08824652e-01,  5.79173560e-01,
        5.82881052e-01,  2.06310467e-01, -8.93941547e-02, -1.58654780e-02,
        3.68055209e-02,  5.86955236e-01,  6.20969331e-01,  2.64735563e-02,
        5.47717735e-01,  3.62460024e-01,  3.46814725e-01,  2.45175573e-01,
        1.92110579e-01,  6.34998684e-01,  7.46654968e-01,  8.47319516e-01,
        5.98530812e-01,  2.48062509e-01,  8.50456825e-01,  8.81932143e-01,
        6.79790132e-01,  5.76903786e-01, -1.39444330e-01,  2.56930870e-01,
        8.51636802e-01,  3.46756932e-01,  7.28507773e-01,  8.78222087e-01,
        5.83931054e-01,  2.84051643e-01,  6.23005049e-01,  7.79185382e-02,
        1.96567494e-01,  6.35400170e-01,  6.52496102e-01,  3.45617031e-01,
        4.89754212e-01,  7.52083282e-01,  6.98281878e-01,  3.64448034e-02,
        3.61535388e-01,  8.72347849e-01,  8.33099379e-01,  8.84805387e-01,
        7.00176299e-01,  6.06765712e-02,  4.47240844e-01,  5.70748463e-01,
        3.96876146e-01, -8.09148539e-02,  5.73609435e-01,  7.66422060e-01,
        7.68695877e-01,  6.74606294e-01,  7.71303536e-01,  2.58587474e-01,
        6.21105076e-01,  6.81693392e-01,  5.96903351e-01,  6.12743771e-01,
        1.28691947e-02, -3.24857518e-02,  4.67748466e-01, -1.61866789e-01,
        6.54692191e-01,  6.53934809e-01,  7.36361510e-01,  3.15447677e-02,
        5.40149563e-01,  8.57436255e-01,  6.15778804e-01,  6.13070478e-01,
        6.46070909e-01,  8.46417996e-01,  5.47356714e-01,  7.17822391e-01,
        3.86712981e-01,  8.74037502e-01,  2.56095098e-01,  2.22823173e-01,
```

```
classifier = Sequential()
```

```
classifier.add(Dense(units=100,activation='relu',input_dim=11))
```

```
classifier.add(Dense(units=50,activation='relu'))
```

```
classifier.add(Dense(units=1,activation='sigmoid'))
```

```
classifier.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

```
y_Pred = linear_regressor.predict(X_test)
```

```
y_Pred
```

```
array([ 7.62887717e-01,  4.17309207e-01,  7.19513090e-01,  7.15054853e-01,
        3.79980456e-01,  1.13424505e-01,  8.39451152e-01,  8.46231370e-01,
        2.84978418e-01,  6.41875966e-01,  8.62406332e-02,  5.99742671e-01,
        4.28457746e-01,  7.15706589e-01,  7.29724974e-01,  6.13714419e-01,
        4.60024696e-01,  4.20990567e-01,  8.96734460e-01,  6.46095902e-01,
        8.62049107e-01,  7.13237038e-01,  2.48144635e-01,  7.32651822e-01,
       -1.85434809e-01,  6.73401457e-01,  6.78942563e-01,  6.92901821e-01,
        5.19398839e-01,  2.70146671e-01,  7.36718111e-01,  8.74465191e-01,
        2.76258230e-01,  4.07425153e-01,  3.56785605e-02,  7.18841330e-01,
        6.91429527e-01,  5.90565303e-01,  5.69260942e-01,  5.24007715e-01,
        2.34339507e-01,  4.48992953e-01,  8.77050640e-01, -1.50655047e-01,
        7.51816236e-01,  7.40139858e-01,  5.34155287e-01,  8.28172811e-01,
        2.71768208e-01,  8.39549293e-01,  1.23728894e-02,  4.07476714e-01,
        6.12313206e-01,  5.74243516e-01,  1.08824652e-01,  5.79173560e-01,
        5.82881052e-01,  2.06310467e-01, -8.93941547e-02, -1.58654780e-02,
        3.68055209e-02,  5.86955236e-01,  6.20969331e-01,  2.64735563e-02,
        5.47717735e-01,  3.62460024e-01,  3.46814725e-01,  2.45175573e-01,
        1.92110579e-01,  6.34998684e-01,  7.46654968e-01,  8.47319516e-01,
        5.98530812e-01,  2.48062509e-01,  8.50456825e-01,  8.81932143e-01,
        6.79790132e-01,  5.76903786e-01, -1.39444330e-01,  2.56930870e-01,
        8.51636802e-01,  3.46756932e-01,  7.28507773e-01,  8.78222087e-01,
        5.83931054e-01,  2.84051643e-01,  6.23005049e-01,  7.79185382e-02,
        1.96567494e-01,  6.35400170e-01,  6.52496102e-01,  3.45617031e-01,
        4.89754212e-01,  7.52083282e-01,  6.98281878e-01,  3.64448034e-02,
        3.61535388e-01,  8.72347849e-01,  8.33099379e-01,  8.84805387e-01,
        7.00176299e-01,  6.06765712e-02,  4.47240844e-01,  5.70748463e-01,
        3.96876146e-01, -8.09148539e-02,  5.73609435e-01,  7.66422060e-01,
        7.68695877e-01,  6.74606294e-01,  7.71303536e-01,  2.58587474e-01,
        6.21105076e-01,  6.81693392e-01,  5.96903351e-01,  6.12743771e-01,
        1.28691947e-02, -3.24857518e-02,  4.67748466e-01, -1.61866789e-01,
        6.54692191e-01,  6.53934809e-01,  7.36361510e-01,  3.15447677e-02,
        5.40149563e-01,  8.57436255e-01,  6.15778804e-01,  6.13070478e-01,
        6.46070909e-01,  8.46417996e-01,  5.47356714e-01,  7.17822391e-01,
```

```
y_Pred = y_Pred.astype(int)
y_Pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```python
print(accuracy_score(y_Pred,y_test))
print("ANN Model")
print("Confusion_Matrix")
print(confusion_matrix(y_test,y_Pred))
print("Classification Report")
print(classification_report(y_test,y_Pred))
```

```
0.4892703862660944
ANN Model
Confusion_Matrix
[[114   0]
 [119   0]]
Classification Report
              precision    recall  f1-score   support

           0       0.49      1.00      0.66       114
           1       0.00      0.00      0.00       119

    accuracy                           0.49       233
   macro avg       0.24      0.50      0.33       233
weighted avg       0.24      0.49      0.32       233
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
from sklearn.model_selection import cross_val_score
```

```python
rf = RandomForestClassifier()
```

```python
rf.fit(X_train,y_train)
ypred = rf.predict(X_test)
```

```python
f1_score(ypred,y_test,average='weighted')
```

```
0.8626963930773419
```

```python
cv = cross_val_score(rf,x,y,cv=5)
```

```python
import pandas as pd
import numpy as np
np.mean(cv)
```

```
0.7801412768226043
```

```
import pandas as pd
import numpy as np
np.mean(cv)
```

```
0.7801412768226043
```

```
import pickle
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from sklearn.model_selection import train_test_split
```

```
import pickle

pickle.dump(XGB,open("rdf.pkl",'wb'))
model = pickle.load(open('rdf.pkl','rb'))
```

```
pickle.dump(model,open('rdf.pkl','wb'))
```