```
import pandas as pd
import numpy as np

df = pd.read_csv("Loan_Data.csv")

df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0

```
df.shape
```

```
(614, 13)
```

```
df.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

```
df.isna().sum()
```

	0
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

```
df.duplicated().sum()
```

```
0
```

```
df.nunique()
```

	0
Loan_ID	614
Gender	2
Married	2
Dependents	4
Education	2
Self_Employed	2
ApplicantIncome	505
CoapplicantIncome	287
LoanAmount	203
Loan_Amount_Term	10
Credit_History	2
Property_Area	3
Loan_Status	2

```
df.columns.tolist()
```

```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area',
 'Loan_Status']
```

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()


df.Gender = le.fit_transform(df.Gender)


df.Married = le.fit_transform(df.Married)


df.Self_Employed = le.fit_transform(df.Self_Employed)


to_be_replaced_with_mode=['Gender','Loan_Amount_Term']
to_be_replaced_with_zero=['Married','Dependents','Self_Employed','Credit_History']


for i in to_be_replaced_with_mode:
    df[i].fillna(df[i].mode(), inplace = True)


for i in to_be_replaced_with_zero:
    df[i].fillna(0,inplace=True)


df.isna().sum()
```

	0
Loan_ID	0
Gender	0
Married	0
Dependents	0
Education	0
Self_Employed	0
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	0
Property_Area	0
Loan_Status	0

```
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace = True)
```

```
df['LoanAmount'].isna().sum()
```

```
0
```

```
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].median(),inplace = True)
```

```
df['Loan_Amount_Term'].isna().sum()
```

```
0
```

```
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	1	0	0	Graduate	0	5849	0.0	146.412162	360.0
1	LP001003	1	1	1	Graduate	0	4583	1508.0	128.000000	360.0
2	LP001005	1	1	0	Graduate	1	3000	0.0	66.000000	360.0
3	LP001006	1	1	0	Not Graduate	0	2583	2358.0	120.000000	360.0

```
df.drop(columns=['Loan_ID'],inplace = True)
```

```
categorical_fields = ['Education','Property_Area','Loan_Status']
for i in categorical_fields:
    df[i] = le.fit_transform(df[i])
```

```
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
0	1	0	0	0	0	5849	0.0	146.412162	360.0	
1	1	1	1	0	0	4583	1508.0	128.000000	360.0	
2	1	1	0	0	1	3000	0.0	66.000000	360.0	
3	1	1	0	1	0	2583	2358.0	120.000000	360.0	
4	1	0	0	0	0	6000	0.0	141.000000	360.0	

```
y=df['Loan_Status']
```

```
df.drop(columns=['Loan_Status'],inplace=True)
```

```
df['Dependents'].replace({'3+':4}, inplace = True)
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(df,y,test_size = 0.2,random_state = 1)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression(random_state = 1).fit(X_train,y_train)
```

```
predictions = classifier.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,predictions)
```

```
0.7804878048780488
```

```
from sklearn.metrics import recall_score
recall_score(y_test,predictions)
```

```
0.9404761904761905
```

```
from sklearn.metrics import precision_score
precision_score(y_test,predictions)
```

```
0.7821782178217822
```

```
from sklearn.metrics import f1_score
f1_score(y_test,predictions)
```

```
0.8540540540540541
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,predictions)
```

```
0.6881868131868132
```

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=150,random_state = 42)
rf.fit(X_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=150, random_state=42)
```

```
accuracy_before = rf.score(X_test,y_test)
```

```
print(f'{accuracy_before:.3f}')
```

```
0.756
```

```
#Extracting feature importances
importances = rf.feature_importances_
feature_names = df.columns
feature_importance_df = pd.DataFrame({'Features':feature_names,'Importance':importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance',ascending = False)
```

```
feature_importance_df
```

	Features	Importance
5	ApplicantIncome	0.221345
7	LoanAmount	0.202928
9	Credit_History	0.167585
6	CoapplicantIncome	0.128142
8	Loan_Amount_Term	0.058137
10	Property_Area	0.052581
2	Dependents	0.052210
0	Gender	0.032867
3	Education	0.028543
4	Self_Employed	0.027841
1	Married	0.027821

```
#Selecting top 8 features
top_features = feature_importance_df['Features'][:8].values
top_features
```

```
array(['ApplicantIncome', 'LoanAmount', 'Credit_History',
       'CoapplicantIncome', 'Loan_Amount_Term', 'Property_Area',
       'Dependents', 'Gender'], dtype=object)
```

```
df
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit
0	1	0	0	0	0	5849	0.0	146.412162	360.0	
1	1	1	1	0	0	4583	1508.0	128.000000	360.0	
2	1	1	0	0	1	3000	0.0	66.000000	360.0	
3	1	1	0	1	0	2583	2358.0	120.000000	360.0	
4	1	0	0	0	0	6000	0.0	141.000000	360.0	
...	...	...	...	...	...	...	...	...	...	
609	0	0	0	0	0	2900	0.0	71.000000	360.0	
610	1	1	4	0	0	4106	0.0	40.000000	180.0	
611	1	1	1	0	0	8072	240.0	253.000000	360.0	
612	1	1	2	0	0	7583	0.0	187.000000	360.0	
613	0	0	0	0	1	4583	0.0	133.000000	360.0	

614 rows × 11 columns

```
for i in feature_names:
    if i not in top_features:
        dff.drop(columns=[i],inplace = True)
```

```
Start coding or generate with AI.
```

```
X_train_selected,X_test_selected,y_train_new,y_test_new = train_test_split(dff,y,test_size = 0.2,random_state = 42)
```

```
X_train_selected = scaler.fit_transform(X_train_selected)
X_test_selected = scaler.transform(X_test_selected)
```

```
classifier_2 = rf.fit(X_train_selected,y_train_new)
```

```
predictions_2 = classifier_2.predict(X_test_selected)
```

```
accuracy_score(y_test_new,predictions_2)
```

```
0.7317073170731707
```

```
accuracy_score(y_test_new,predictions_2)
```

```
0.7317073170731707
```

```
dff = df.copy(deep=True)
```

```
dff
```

	Gender	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1	0	5849	0.0	146.412162	360.0	1.0	2
1	1	1	4583	1508.0	128.000000	360.0	1.0	0
2	1	0	3000	0.0	66.000000	360.0	1.0	2
3	1	0	2583	2358.0	120.000000	360.0	1.0	2
4	1	0	6000	0.0	141.000000	360.0	1.0	2
...	...	...	...	...	...	...	...	...
609	0	0	2900	0.0	71.000000	360.0	1.0	0
610	1	4	4106	0.0	40.000000	180.0	1.0	0
611	1	1	8072	240.0	253.000000	360.0	1.0	2
612	1	2	7583	0.0	187.000000	360.0	1.0	2
613	0	0	4583	0.0	133.000000	360.0	0.0	1

614 rows × 8 columns

```
from sklearn.neural_network import MLPClassifier
```

```
clf = MLPClassifier(hidden_layer_sizes = (64,32,16), activation = 'logistic', solver = 'adam', max_iter = 500, random_state = 42 )
```

```
clf.fit(X_train,y_train)
```

```
MLPClassifier(activation='logistic', hidden_layer_sizes=(64, 32, 16),
              max_iter=500, random_state=42)
```

```
predictions_3 = clf.predict(X_test)
```

```
accuracy_score(y_test,predictions_3)
```

```
0.7804878048780488
```