LOAN APPLICATION APPROVAL STATUS

Loans are no longer considered a last resort to buy a sought-after smart phone or a dream house. Over the last decade or so, people have become less hesitant in applying for a loan, whether it is personal, vehicle, education, business, or home – especially when they don't have a lump sum at their disposal. It also helps that banks and other financial institutions are making it easier for customers and prospective borrowers to get loans with minimal paperwork, quick eligibility checks, and competitive interest rates.

As there is a direct relationship between risk and reward and the quest for profit maximization has given rise to accelerated risk taking for enhanced rewards. Henceforth to increase the business of the bank, banks are trying to get a large number of loan applications. With increase in applications, risk associated with it is also increasing day by day. Credit risk is the biggest risk for banks or financial institutions. It occurs when borrowers or counterparties fail to meet contractual obligations.

While banks cannot be fully protected from credit risk due to the nature of their business model. However, they can lower their exposure in several ways. To lower their risk exposure, they can loan money to people with good credit histories, transact with high-quality counterparties, or own collateral to back up the loans. The decision about whether to grant credit to a certain customer must be evaluated on a case-to-case basis. Each small business that grapples with this issue needs to gather and evaluate financial information, decide whether to grant credit and if so how much. In this situation issue of Credit Risk assessment arises, which helps them to evaluate if a loan applicant could be a defaulter at a later stage, whether they could go ahead and grant the loan or not. This helps the banks to minimize the possible losses and can increase the volume of credits.

It is very difficult to predict the possibility of repayment of loan by the customer. In recent years many researchers worked on loan approval prediction systems. Load Application Status Prediction is a task that can be done based on historical information of the customer and bank. Analysing the dataset already existed regarding the status of the Loan Application and creating a model, could help financial institutions to predict the status of Loan Application.

A Prediction Model uses data mining, statistics and probability to forecast an outcome. Every model has some variables known as predictors that are likely to influence future results. The data that was collected from various resources then used to make a statistical model. As big the training as good the predictive model would be.

Here various machine learning algorithms will be used on existing dataset to extract important insides and make a model to predict nearly perfect decision to approve or to reject the loan request of a particular customer.

Dataset includes the following details of applicants who have applied for loan:

1. Loan_ID : It is simply a unique application number of the customer.
2. Gender : Sex of the applicant.
3. Married : Marital status of the applicant.
4. Dependents : Number of members in the family, who are dependent on the applicant.
5. Education : Educational qualification of the applicant.
6. Self_Employed : Employment status of the applicant. It is important because repayment capacity depends on the income of the person.
7. ApplicantIncome : Apllicant income is important to calculate EMI to Income Ratio.
8. CoapplicantIncome : A co-applicant is a person who joins in the application of a loan. Having a co-applicant can increase the chances of approval of the application.
9. LoanAmount : Amount of loan applied by the applicant.
10. Loan_Amount_Term : The shorter the repayment period, the more credible would be the applicant.
11. Credit_History : A credit score tells a lot about applicants financial health. Whether applicant paid his/her EMIs on time or not.
12. Property_Area : Area of the property is urban or rural.
13. Loan_Status : Status of the application (Approved or rejected). It is the Target variable here.

Translate Business Problem into Machine Learning problem :

This is a classification problem where we have to predict whether a loan will be approved or not. Specifically, it is a binary classification where we have to predict either one of the two classes given i.e. approved (Y) or not approved (N). Another way to frame the problem is to predict whether the loan will likely to default or not, if it is likely to default, then the loan would not be approved, and vice versa. The dependent variable or target variable is the Loan_Status, while the rest are independent variable or features. We need to develop a model using the predictors to predict the target variable.

In [1]:
```
# IMPORTING THE REQUIRED LIBRARIES:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# importing the dataset:

dataset = pd.read_csv("https://raw.githubusercontent.com/dsrscientist/DSData/master/loan_prediction.csv")
dataset.head()
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [3]:
```
# checking the shape of the dataset:

dataset.shape
```

Out[3]:
```
(614, 13)
```

In [ ]:
```
# Checking the information about the data:

dataset.info()
```

- From the above output, we observe that Gender, Married, Dependents, Self_Employed, LoanAmount, Loan_Amount_Term, Credit_History are having NULL values.

-In total there are 13 columns out of which only 5 are numeric and others 8 are categorical variables.

Number of Missing values of the columns:

Gender: 13, Married: 3, Dependents: 15, Self_Employed: 32, LoanAmount: 22, Loan_Amount_Term: 14, Credit_History: 50

Where, Gender, Married, Self_Employed are categorical variables and are imputed by their respective mode.

And LoanAmount, Loan_Amount_Term and Credit_History are numerical variables are imputed by their respective means.

First, removing missing values of categorical columns :

In [5]:
```
# 1. Gender :

dataset['Gender'].value_counts()
```

Out[5]:
```
Male      489
Female    112
Name: Gender, dtype: int64
```

In [6]:
```
dataset['Gender'] = dataset.Gender.fillna('Male')
```

In [7]:
```
# 2. Married, is a categorical column so missing value would be replaced by its mode :

dataset['Married'].value_counts()
```

Out[7]:
```
Yes    398
No     213
Name: Married, dtype: int64
```

In [8]:
```
dataset['Married'] = dataset.Married.fillna('Yes')
```

In [9]:
```
# 3. Self_Employed

dataset['Self_Employed'].value_counts()
```

Out[9]:
```
No     500
Yes     82
Name: Self_Employed, dtype: int64
```

In [10]:
```
dataset['Self_Employed'] = dataset.Self_Employed.fillna('No')
```

Now, Removing missing values of numerical columns:

In [11]:
```
# Removing missing values of LoanAmount by its mean:

dataset.LoanAmount = dataset.LoanAmount.fillna(dataset.LoanAmount.mean())
```

In [12]:
```
# Removing missing values of Loan_Amount_Term by its mean:

dataset['Loan_Amount_Term'] = dataset["Loan_Amount_Term"].fillna(dataset['Loan_Amount_Term'].mean())
```

In [13]:
```
#  Dependents, if we see number of dependents should be an integer, however in our dataset its an 'Object' data type:

dataset['Dependents'].value_counts()
```

Out[13]:
```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

Here, Dependents are having some values as 3+ which means an individual is having more than 3 dependents.

In [14]:
```
# With '+' sign we cannot move ahead, so remove the + sign and consider the number of dependents as 3.

dataset['Dependents'] = dataset['Dependents'].str.replace(r'\D', '')
```

In [15]:
```
dataset['Dependents'].value_counts()
```

Out[15]:
```
0    345
1    102
2    101
3     51
Name: Dependents, dtype: int64
```

In [16]:
```
# changing the data type of Dependents column:

dataset["Dependents"] = dataset["Dependents"].astype(str).astype(float)
```

In [17]:
```
dataset['Dependents'] .isnull().sum()
```

Out[17]:
```
15
```

In [18]:
```
dataset['Dependents'] = dataset['Dependents'].fillna(dataset['Dependents'].mean())
```

In [19]:
```
# As number of dependents cannot be in float, hence changing the type to int:

dataset['Dependents'] = dataset['Dependents'].astype(int)
```

In [20]:
```
dataset['Dependents'].value_counts()
```

Out[20]:
```
0    360
1    102
2    101
3     51
Name: Dependents, dtype: int64
```

In [21]:
```
# Credit_History

dataset['Credit_History'].value_counts()
```

Out[21]:
```
1.0    475
0.0     89
Name: Credit_History, dtype: int64
```

In [ ]:
```
dataset['Credit_History'].isnull().sum()
```

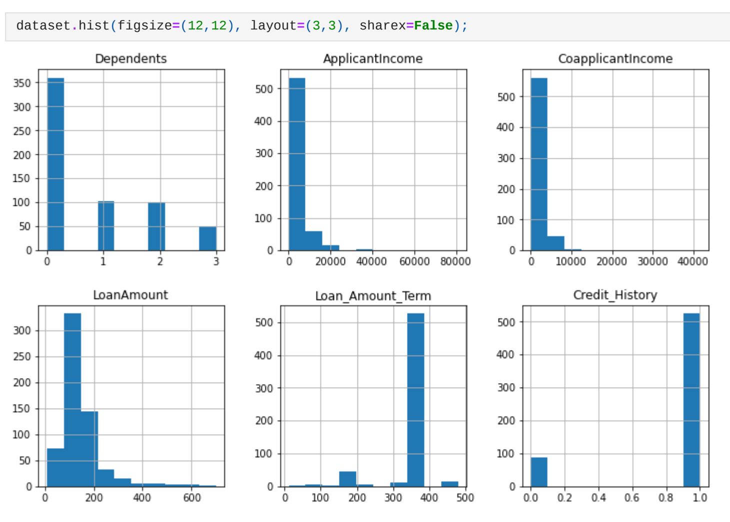
In [22]:
```
dataset['Credit_History'] = dataset.Credit_History.fillna(1.0)
```

SO, WE HAVE SUCCESSFULLY REMOVED ALL THE MISSING VALUES.

Now, checking for the OUTLIERS:

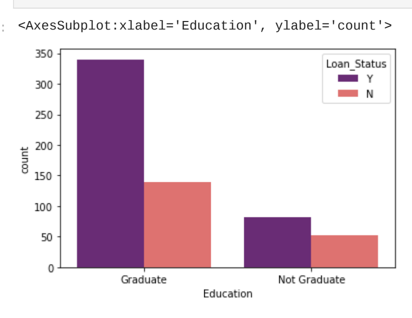
BOXPLOT

In [23]:
```
dataset.hist(figsize=(12,12), layout=(3,3), sharex=False);
```



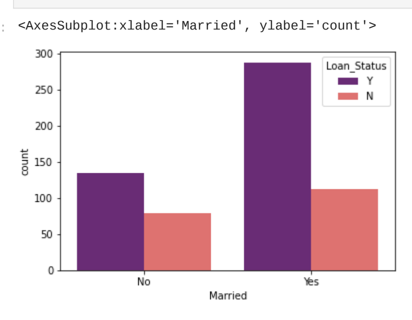
In [24]:
```
sns.countplot(dataset['Education'], hue = 'Loan_Status', data = dataset, palette = 'magma')
```

Out[24]: <AxesSubplot:xlabel='Education', ylabel='count'>



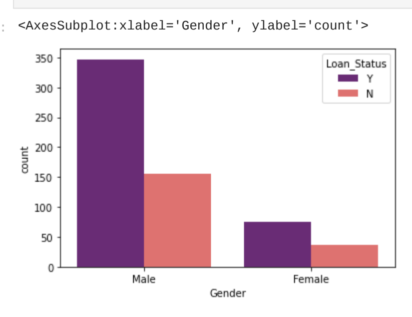
In [25]:
```
sns.countplot(dataset['Married'], hue = 'Loan_Status', data = dataset, palette = 'magma')
```

Out[25]: <AxesSubplot:xlabel='Married', ylabel='count'>



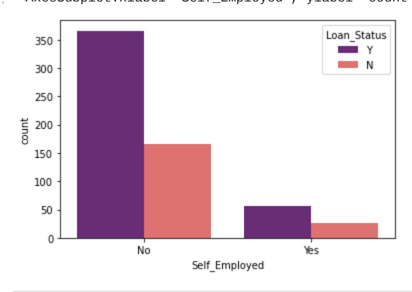
In [26]:
```
sns.countplot(dataset['Gender'], hue = 'Loan_Status', data = dataset, palette = 'magma')
```

Out[26]: <AxesSubplot:xlabel='Gender', ylabel='count'>



In [27]:
```
sns.countplot(dataset['Self_Employed'], hue = 'Loan_Status', data = dataset, palette = 'magma')
```

Out[27]: <AxesSubplot:xlabel='Self_Employed', ylabel='count'>

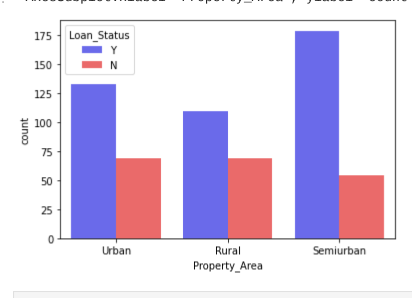


In [28]:
```
sns.countplot(dataset['Credit_History'], hue = 'Loan_Status', data = dataset, palette = 'magma')
```

Out[28]: <AxesSubplot:xlabel='Credit_History', ylabel='count'>

In [29]:
```
sns.countplot(dataset['Property_Area'], hue = 'Loan_Status', data = dataset, palette = 'seismic')
```

Out[29]: <AxesSubplot:xlabel='Property_Area', ylabel='count'>

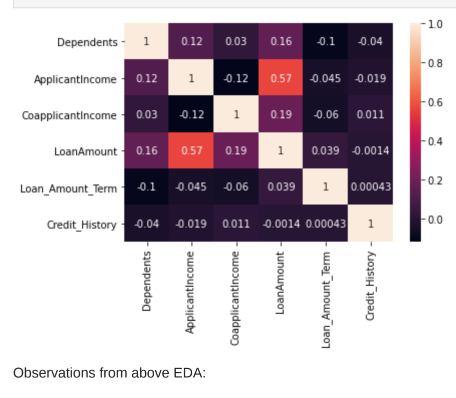


In [30]:
```
# checking the correlation between variables:

dataset.corr()

# HEATMAP

sns.heatmap(dataset.corr(), annot=True);
```



Observations from above EDA:

1.There is not a substantial difference between male and female approval rates. 2.Married applicants have a slightly higher chances of loan approval. 3.Graduates have higher chance of loan approval compared to non-graduates. 4.Self_Employed employees have slightly lower chances of loan approval. 5.People with credit history as 1 are more likely to get their loans approved. 6.Propotion of loans getting approved in semiurban area is higher as compared to that in rural or urban areas. 7.We see that the most correlated variables are (ApplicantIncome - LoanAmount) and (Credit_History - Loan_Status). LoanAmount is also correlated with CoapplicantIncome

In [31]:
```
# CONVERTING CATEGORICAL VARIABLES INTO NUMERIC :
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

Categorical_feature = ['Gender','Education', 'Married', 'Self_Employed', 'Property_Area', 'Loan_Status']
for i in Categorical_feature:
    dataset[i] = le.fit_transform(dataset[i])
```

In [32]:
```
# FEATURE SELECTION:
X = dataset.iloc[:, 1:12]    # Loan id is not of use for the study
Y = dataset.iloc[:, 12]
```

In [33]:
```
#now, we will split the dataset into two parts for training and testing in the ratio of 80:20 -

X_train,X_test,Y_train,Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0 )
```

Model Fitting:

1.DecisionTreeClassifier 2.Logistic Regression 3.Random Forest Classifier

In [ ]:
```
#1. DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier

# creating Decision Tree classifier object:

model = DecisionTreeClassifier()

# Train Decision Tree Classifier

model.fit(X_train,Y_train)

# prediction of Y_test

Y_pred = model.predict(X_test)
Y_pred
```

In [36]:
```
# Model Evaluation
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(Y_test, Y_pred)
cnf_matrix
```

Out[36]:
```
array([[19, 14],
       [27, 63]], dtype=int64)
```

In [37]:
```
print("Accuracy:", metrics.accuracy_score(Y_test, Y_pred))
```

```
Accuracy: 0.6666666666666666
```

Accuracy: 0.6666666666666666 So, we got the accuracy of DecisionTreeClassifier as 67% .

Now, we will check for another model with better accuracy score.

In [ ]:
```
# Fitting Model 2. Logistic Regression

logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, Y_train)
Y_predict = logistic_regression.predict(X_test)
Y_predict
```

In [39]:
```
# Model Evaluation using Confusion matrix
cnf_matrix = metrics.confusion_matrix(Y_test, Y_predict)
cnf_matrix
```

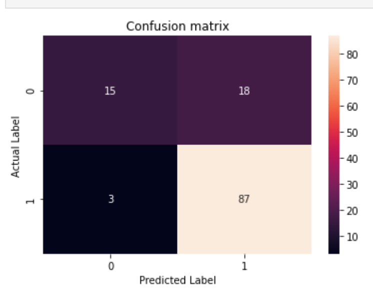
Out[39]:
```
array([[15, 18],
       [ 3, 87]], dtype=int64)
```

In [40]:
```
print("Accuracy:", metrics.accuracy_score(Y_test, Y_predict))
```

```
Accuracy: 0.8292682926829268
```

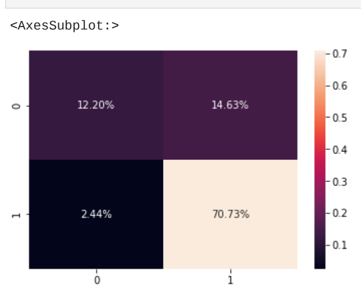
Here, we get the accuracy rate as 82.93% which is much better than DecisionTreeClassifier.

In [41]:
```
#Visualization of Confusion Matrix using Heatmap
```



In [42]:
```
sns.heatmap(cnf_matrix/np.sum(cnf_matrix), annot=True, fmt = '.2%')
```

Out[42]: <AxesSubplot:>



So, for Loan Application status prediction :

Y_predict = logistic_regression.predict(X_test)

model gives fairly significant results.

In [43]:
```
# # Fitting Model 3. Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()

model1 = rfc.fit(X_train,Y_train)

Y_predict2 = model1.predict(X_test)
Y_predict2
print("Accuracy:", metrics.accuracy_score(Y_test, Y_predict2))
```

```
Accuracy: 0.7886178861788617
```

In [44]:
```
cnf_matrix = metrics.confusion_matrix(Y_test, Y_predict2)
cnf_matrix
```

Out[44]:
```
array([[14, 19],
       [ 7, 83]], dtype=int64)
```

From above all analysis we find that LogisticRegrssion Model gets the highest accuracy rate as 82.93% .

Conclusion:

Logistics Regression model gives the best accuracy out of DecisionTreeClassifier,Logistic Regression,Random Forest Classifier models.

Future Work:

```
- Model Parameter Tuning and Selection
- Finding ROC AUC score
- Training 70% data and testing on 30%
```