

Chapter I

TOOLS AND TECHNOLOGY USED IN PROJECT

2.1 Language used :

2.1.1 python :

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

To learn more about python goto: <https://www.python.org/>

2.1.2 Python version used:

In the project and training, I used python version 3.7.2 .Python 3.7.2 is the second maintenance release of Python 3.7. The Python 3.7 series is the newest major release of the Python language and contains many new features and optimizations.

2.2 Tools used :

2.2.1 Anaconda:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 15 million users and includes more than 1500 popular data-science packages suitable for Windows, Linux, and MacOS

2.2.2 Jupyter notebook

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ``.ipynb`` extension.

2.2.3 Google colaboratory

I have also used Google colaboratory for implementing Machine learning during my training. Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.

2.3 packages used:

2.3.1 NumPy

NumPy is an open source library available in Python that aids in mathematical, scientific, engineering, and data science programming. NumPy is an incredible library to perform mathematical and statistical operations. It works perfectly well for multi-dimensional arrays

2.3.2 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

2.3.3 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

2.3.4 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

2.4 Technology learned:

2.4.1 Machine learning :

Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Or in more simpler way we can explain machine learning as :

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed."

This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Machine learning problem can be assigned to one of **two broad classifications**:

2.4.1.1 supervised learning:

supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each

example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances.

Supervised learning problems are categorized into:

- Regression

In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function

- Classification

In here, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories. or in other words we can say that in classification we are just predicting the value of the class to which the input data belong

2.4.1.2 Unsupervised learning.

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data

2.4.2 Machine learning models:

There are various types of machine learning models based on the types of the time of machine learning (i.e. – supervised , unsupervised)

- A model is the relationship between features and the label. (Tensorflow)
- An ML model is a mathematical model that generates predictions by finding patterns in your data. (AWS ML Models)
- ML Models generate predictions using the patterns extracted from the input data (Amazon Machine learning)

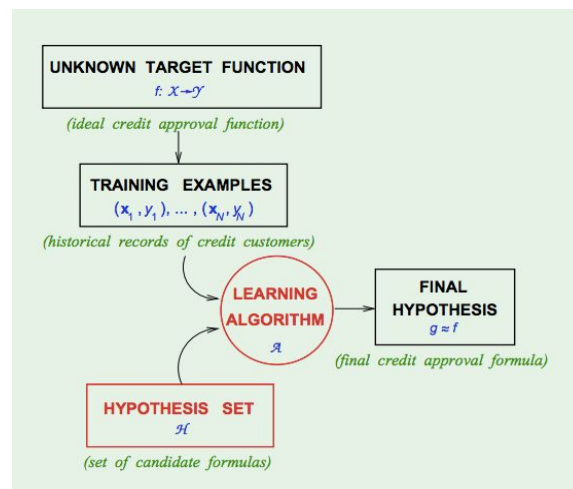


fig :machine learning flow model

2.4.2.1 Regression models:

2.4.2.1.1 Linear regression:

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

2.4.2.2 Classification model:

2.4.2.2.1 Logistic regression:

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary classification).

2.4.2.2.2 K-nearest neighbour:

the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

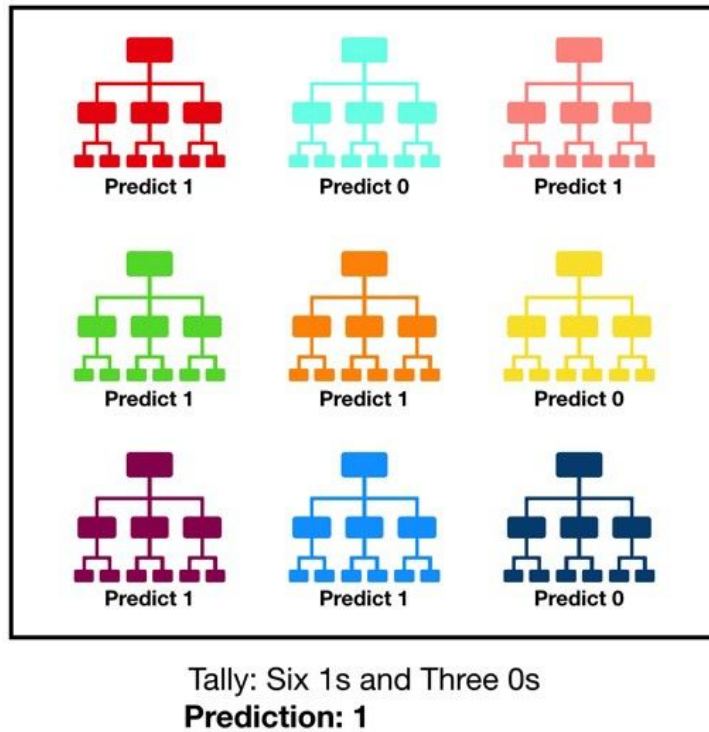
In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

2.4.2.2.3 Random forest classifier:

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction (see figure).

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds.



2.4.2.2.4 Decision tree:

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Chapter II

Demonstration of technology through project

3.1 Problem Statement:

Multiclass classification problem for predicting loan status from a rich dataset. The problem can be reduced to a binary classification problem to build a loan approval pre-check system for potential customers.

3.2 A potential solution to problem:

3.2.1 Loan status foreteller:

In finance, a loan is the lending of money by one or more individuals, organizations, or other entities to other individuals, organizations etc. The recipient (i.e. the borrower) incurs a debt, and is usually liable to pay interest on that debt until it is repaid, and also to repay the principal amount borrowed.

Now a days there is a process which is called credit score checker which takes your information of past histories of any loan or any unpaid amount left in previous loans and give you a number based on many thing mentioned above and many more (like income of applicant etc.).

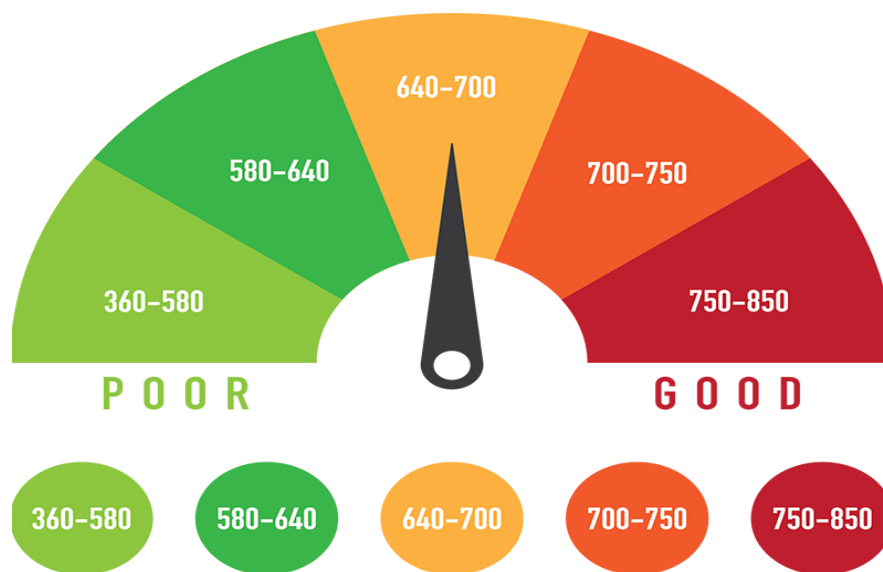


Fig – credit scores demonstration

There are many sites available that can do. What this loan status foreteller does is that it takes your information as a input to its trained machine learning model on the basis of the training data it took. Now after that it will forecast whether you are eligible for this loan or not. It will tell you that the amount you want and according to your income and the rate of repayment is under the criteria of a bank or not.

The major proportion of the companies like policy bazar are spending their money on these type of projects as they can attract the customer by telling them the status of their loan beforehand ,so saving the precious time of their customer by saving all the hectic running around in banks for loans.

3.2.2 Things on which your loan status depends:

3.2.2.1 Credit history:

Your credit history is indicative of your future repayment behaviour, based on your pattern in settling past loans. It helps the bank to know if you will be punctual and regular with your payments. Any default or delay in the past is investigated – the longer the delay, the lower your score is likely to be

Generally, a credit score between 700 and 800 is positive. That means you are likely to be favoured as a safe applicant with a clean history devoid of any repayment defaults. On the other hand, credit score of less than 300 will increase the likelihood of your application being rejected. Specialised bureaus such as CIBIL are a source of credit scores that banks seek information from to assess your creditworthiness.

3.2.2.2 Work experience:

Banks weigh your employment history and current engagement to ensure that your source of income is reliable. A bank wants to be certain that your employer is financially sound, with no history of outstanding or delay in paying employees their salaries. Stability of

your job matters too. Therefore, government jobs have the added advantage of being perceived as safe compared to lesser known private companies or self-employment.

The idea is that your capacity for repaying the loan depends on your income, so its source needs to be reliable and consistent. Banks prefer applicants who have worked longer in their present employment, as it also establishes stability.

3.2.2.3 Age:

Your age matters because it is indicative of your financial stability. You start working in your 20s and by the time you turn 30 you would have five or six years of work experience. So you are financially stable and moving up the proverbial corporate ladder with a better salary. As you progress further in the next 20 or 30-odd years you will have fewer earning years to repay your loans. Therefore, a loan application in your retirement years is likely to be rejected.

3.2.2.4 Income:

As already mentioned, your income represents your repayment capacity. Banks assess your income capacity in the backdrop of existing debt obligations, dependents, source, and duration. In this context, one of the many things the bank checks is sufficient surplus after EMI payments. If this is found wanting, the bank infers that you're spread far too thin and likely to default. However, if the ratio is five times and above, the bank will consider you financially healthy.

Your eligibility gets better if you can show additional sources of income, such as your spouse's salary. This indicates better repayment capacity as you have more than one source of income to tap into

3.2.2.5 Repayment:

If you choose a shorter repayment period, you have a better chance of getting the loan approved.

Several banks favour applications for a repayment period of up to five years. As the repayment period increases in five-year slabs – 10, 15, 20, and 25 years – the score reduces.

3.3 Literature Survey:

Loan default trends have been long studied from a socio-economic standpoint. Most economic papers believe in empirical modelling of these complex systems in order to be able to predict the loan default rate for a particular individual. The use of machine learning for such tasks is a trend which we are observing now. Some of the papers to understand the past and present perspective of loan defaults are

- Serrano-Cinca C, Gutierrez-Nieto B, Lopez Palacios L (2015), Determinants of Default inP2P Lending. PLoS ONE 10(10)

URL: <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0139427>

- A.R.Ghatge, P.P.Halkarnikar develops the artificial neural network model for predict the credit risk of a bank. The Feed- forward back propagation neural network is used to forecast the credit default. They also compare the results with the manual calculations of the bank conducted in year 2004, 2005 and 2006. The results give the better and higher performance over manual calculations of bank
- Maher Alaraj, Maysam Abbod, and Ziad Hunaiti proposed a new ensemble method for classification of consumer loan. This ensemble method is based on neural networks. They state that the proposed method give better results and accuracy as compared to single classifier and any other model.
- Marc Claesen, Frank De Smet, Johan A.K. Suykens, Bart De Moor[8] proposed a model based on support vector machine that reduced the complexity of training data and

redicts the model with high accuracy. This model is used to avoid duplicate storage of data.

3.4 About dataset:

Each row represent a customer, each columns contains applicant's attributes described on the columns metadata. The raw data contains 614 rows (applicant) and 13 columns (features). The loan status is our target variable i.e. which we will predict.

3.4.1 Columns:

- **Loan_ID:** Loan ID
- **Gender:** Gender of the applicant
- **Married:** Marital status of applicant
- **Dependents:** number of dependents on applicant i.e.-number of people that are dependent on the applicant.
- **Education:** whether the applicant is graduate or non-graduate.
- **Self_Employed:** whether the applicant is self-employed or not.
- **ApplicantIncome:** applicant's monthly income.
- **CoapplicantIncome:** additional sources of income, such as your spouse's income.
- **LoanAmount:** Loan amount which applicant want for his or her needs.
- **Loan_Amount_Term:** the repayment term of loan in days.
- **Credit_History:** whether or not the applicant holds previous credit history or not.
- **Property_Area:** the area of property applicant to put collateral for bank.
- **Loan_Status:** based on the above mentioned attributes whether your loan application would get approved or not.

3.5 Working of the project

My whole project is based on the classification problem that is retrieved from the problem statement. Now for classification of the status of the loan is a binary classification there are plenty of algorithms that can do classification but here for the scope of this project report only three machine learning models will be sufficient ,naming :

1. K-nearest neighbor
2. Random forest classifier
3. Logistic regression

The basic concept of these algorithms mentioned are and given in chapter 2 of this project report and from this point, this project report will be oriented towards understanding these machine learning models how they work, what are their pros and cons. comparing these three models we will conclude that which model will be best for these type of small level analysis. The steps related to this project are:

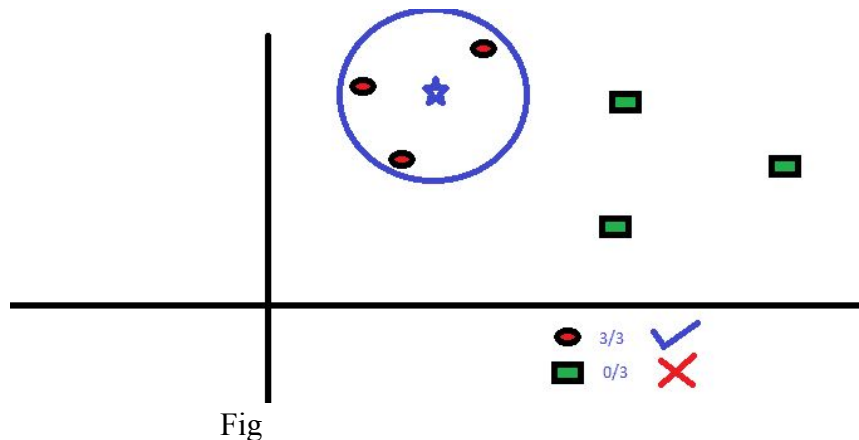
1. Data extraction
2. Missing value detection
3. Missing value handling
4. Label encoding of the categorical values
5. Visualization
6. Outlier detection
7. Outlier handling
8. Finding the optimal parameters for Applying the ml model
9. Applying and comparing the results of the model
10. Conclusion of the model analysis

3.5.1 Working of k-nearest neighbour

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS):

Fig

You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The “K” in KNN algorithm is the nearest neighbors we wish to take vote from. Let's say $K = 3$. Hence, we will now make a circle with BS as centre just as big as to enclose only three data points on the plane. Refer to following diagram for more details:



The three closest points to BS are all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbour went to RC. The choice of the parameter K is very crucial in this algorithm. Next we will understand what are the factors to be considered to conclude the best K.

Now in the loan prediction project the knn algorithm works similar to the above mentioned example the knn takes the Euclidian distance of the k nearest neighbours and give the prediction on the factor that the maximum no. of the neighbour belongs to which category.

3.5.2 Working of random forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Now in perspective of our project the random forest works by taking different features as the branches of its decision trees, each decision tree will make the prediction of the present input and give the output and then the main prediction will be the one which majority of the trees have predicted hence the category of the loan status will depend decided on the

population that predict the value. So this model can be a potential model for our problem.

3.5.3 Working of logistic regression

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function.

We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given x . However, it is easy to construct examples where this method performs very poorly. Intuitively, it also doesn't make sense for $h_{\theta}(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$. To fix this, let's change the form for our hypotheses $h_{\theta}(x)$ to satisfy $0 \leq h_{\theta}(x) \leq 1$. This is accomplished by plugging $\theta^T x$ into the Logistic Function.

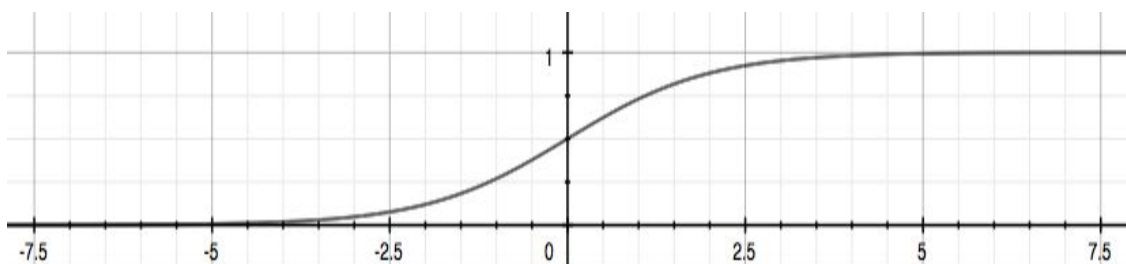
Our new form uses the "Sigmoid Function," also called the "Logistic Function":

$$h_{\theta}(x) = g(\theta^T x)$$

$$z = \theta^T x$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

The following image shows us what the sigmoid function looks like:



Fig

The function $g(z)$, shown here, maps any real number to the $(0, 1)$ interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification.

$h_{\theta}(x)$ will give us the probability that our output is 1. For example, $h_{\theta}(x)=0.7$ gives us a probability of 70% that our output is 1. Our probability that our prediction is 0 is just

the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

$$h_{\theta}(x)=P(y=1|x;\theta)=1-P(y=0|x;\theta)$$

$$P(y=0|x;\theta)+P(y=1|x;\theta)=1$$

3.6 Assumptions

- We have not made any direct assumptions about our data or models as per our best knowledge.

3.7 Limitations

- We have not considered the effect of delayed payments which can't be derived from various fields of the dataset.
- Cross-validation for support vector machine modelled here requires high computational time.

3.8 Evaluation of all the models

All the models will be evaluated on the basis of how they perform on the testing set data and cross validation data. There are various methods for comparing the models according to their accuracy score according to their roc curve and auc etc.

CHAPTER-IV

screenshots of the project

4.1 Demonstration through project:

4.1.1 Extraction of data:

```
>>import pandas as pd

>>import numpy as np

>>import seaborn as sns

>>from matplotlib import pyplot as plt

>>data =

    pd.read_csv('https://raw.githubusercontent.com/PayelGanguly96/Loan-
    Prediction/master/train_loanPrediction.csv')

>>data.head()
```

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

Fig

```
>> data.shape
```

```
(614, 13)
```

This dataset contains 614 rows and 13 columns.

```
>> data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 605 entries, 0 to 613
Data columns (total 13 columns):
 Loan_ID                605 non-null object
 Gender                 605 non-null int64
 Married                605 non-null int64
 Dependents             605 non-null int64
 Education              605 non-null int64
 Self_Employed          605 non-null int64
 ApplicantIncome         605 non-null int64
 CoapplicantIncome       605 non-null float64
 LoanAmount              605 non-null float64
 Loan_Amount_Term        605 non-null float64
 Credit_History          605 non-null int64
 Property_Area           605 non-null int64
 Loan_Status             605 non-null int64
dtypes: float64(3), int64(9), object(1)
memory usage: 66.2+ KB
```

fig

There are 8 object columns , one integer column, and 4 floating point columns.

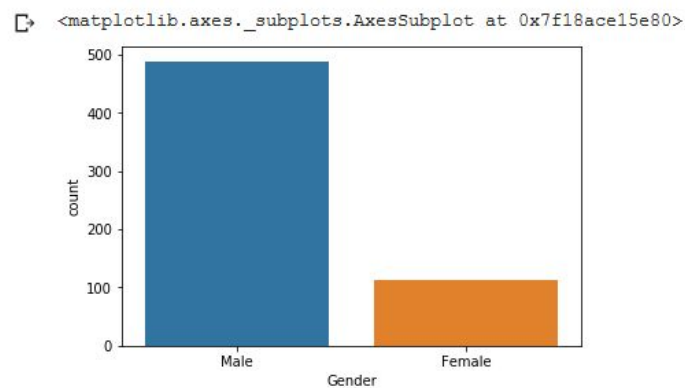
```
>> data.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

Fig

4.1.2 Data visualization (exploratory data analysis):

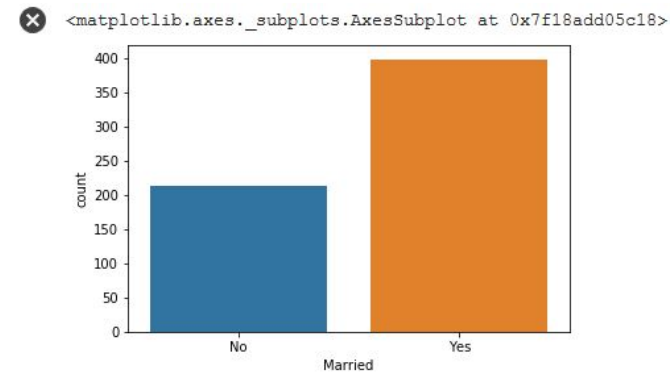
```
>> sns.countplot(data.Gender)
```



Fig

here we can see that the proportion of male population are comparatively much higher in number than womens this histogram will help us later when dealing with the missing values.

```
>> sns.countplot(data.Married)
```



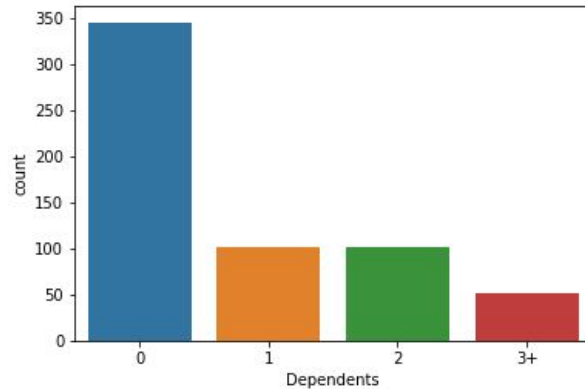
Fig

This histogram shows that mostly applicants who are applying for the loan are married.this will also help us after some time.

```
>> sns.countplot(data.Dependents)
```



```
<matplotlib.axes._subplots.AxesSubplot at 0x7f18ad26aba8>
```



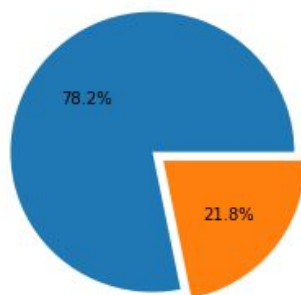
Fig

In this graph as we all can see that the majority of the applicants does not have many non-earning dependents

```
>> plt.pie(data.Education.value_counts(), explode=(0, 0.1), autopct='%1.1f%%')
```


Fig

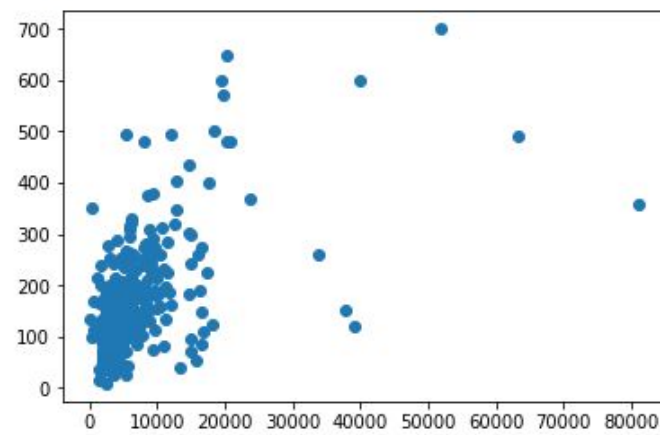
```
([<matplotlib.patches.Wedge at 0x7f18a86ca400>,  
<matplotlib.patches.Wedge at 0x7f18a86cab00>],  
[Text(-0.8514262161117528, 0.6964721089301588, ''),  
Text(0.928828599394639, -0.7597877551965374, '')],  
[Text(-0.4644142996973196, 0.37989387759826837, '78.2%'),  
Text(0.541816682980206, -0.44320952386464674, '21.8%')])
```



here in this pie plot we can see that the most applicants are graduate so we can say that this imparts that the educated ones are in majority that take loans.

```
>> plt.scatter(data.ApplicantIncome,data.LoanAmount)
```

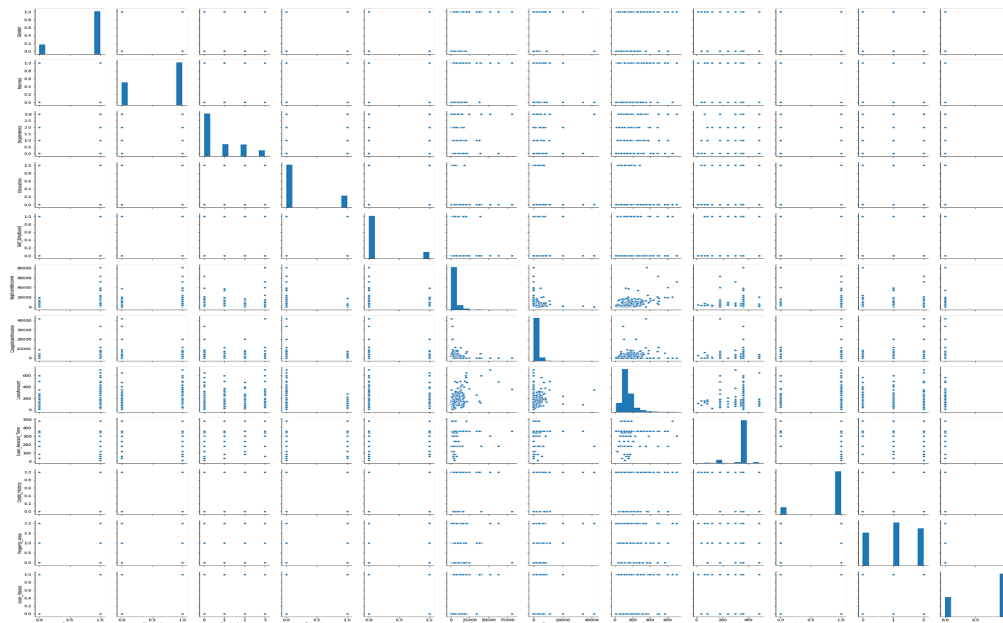
 <matplotlib.collections.PathCollection at 0x7f18a880b438>



Fig

as we can see in this scatter plot that there are some outliers that can ruin the machine learning model's prediction accuracy. so we have to remove them

```
>> sns.pairplot(data)
```



fig

This above given figure is an all in one data visualization called pair plot i.e.- there is every possible combination of attributes plotted on graphs and on histogram as and when required.

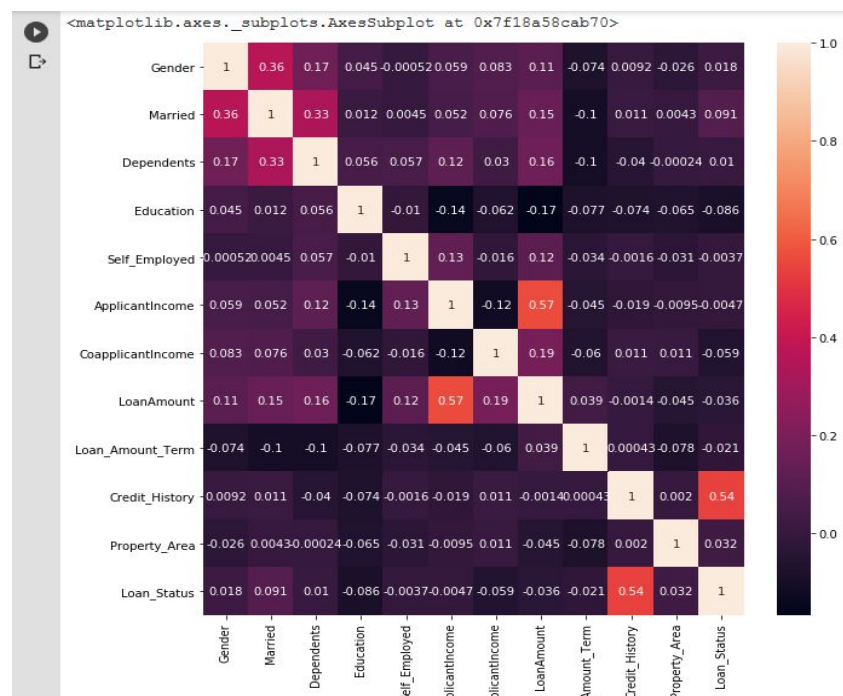
4.1.3 Finding relation between data itself:

```
>> data.corr()
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
Gender	1.000000	0.364569	0.172914	0.045364	-0.000525	0.058809	0.082912	0.107930	-0.073567	0.0091
Married	0.364569	1.000000	0.334216	0.012304	0.004489	0.051708	0.075948	0.147141	-0.100863	0.0109
Dependents	0.172914	0.334216	1.000000	0.055752	0.056798	0.118202	0.030430	0.163106	-0.101054	-0.0401
Education	0.045364	0.012304	0.055752	1.000000	-0.010383	-0.140760	-0.062290	-0.166998	-0.077242	-0.0736
Self_Employed	-0.000525	0.004489	0.056798	-0.010383	1.000000	0.127180	-0.016100	0.115260	-0.033943	-0.0015
ApplicantIncome	0.058809	0.051708	0.118202	-0.140760	0.127180	1.000000	-0.116605	0.565620	-0.045242	-0.0186
CoapplicantIncome	0.082912	0.075948	0.030430	-0.062290	-0.016100	-0.116605	1.000000	0.187828	-0.059675	0.0111
LoanAmount	0.107930	0.147141	0.163106	-0.166998	0.115260	0.565620	0.187828	1.000000	0.038801	-0.0014
Loan_Amount_Term	-0.073567	-0.100863	-0.101054	-0.077242	-0.033943	-0.045242	-0.059675	0.038801	1.000000	0.0004
Credit_History	0.009170	0.010938	-0.040160	-0.073658	-0.001550	-0.018615	0.011134	-0.001431	0.000432	1.0000
Property_Area	-0.025752	0.004257	-0.000244	-0.065243	-0.030860	-0.009500	0.010522	-0.044776	-0.077620	0.0019
Loan_Status	0.017987	0.091478	0.010118	-0.085884	-0.003700	-0.004710	-0.059187	-0.036416	-0.020974	0.5405

```
>> plt.figure(figsize=(10,10))
```

```
>> sns.heatmap(data.corr(),annot=True)
```

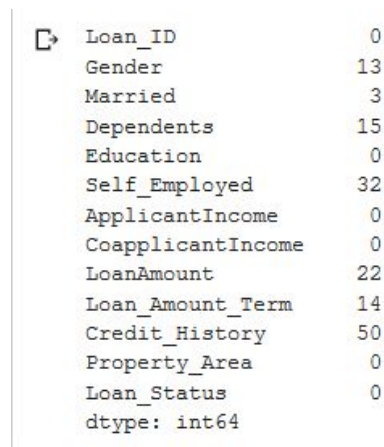


Fig

As we can say that the heat map is much more easy to interpret for so many values and data features than the correlation table.

4.1.4 Checking for missing values:

```
>> data.isna().sum()
```



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

Fig

as we can see in the output that there are many columns which are having nan values .so to deal with this data we have to remove these nan values .

4.1.5 Removal of missing values:

imputation is the process of replacing missing data with substituted values. We do our imputation using the imputer class of sklearn's preprocessing module.

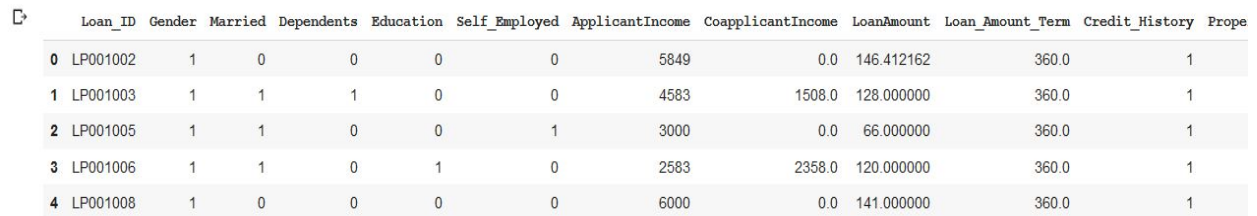
```
>> im=Imputer()  
  
>>for j in ['LoanAmount', 'Loan_Amount_Term']:  
    data[j]=im.fit_transform(data[[j]])
```

this code here will deal with the missing values of object type attribute

```
>> for x in ['Gender', 'Married', 'Dependents',
            'Education', 'Self_Employed', 'Property_Area',
            'Loan_Status', 'Credit_History']:
>>     data[x].fillna(value=data[x].value_counts().index[0], inplace=True)
```

4.1.6 Label encoding:

```
>> le=LabelEncoder()
>> for x in ['Gender', 'Married', 'Dependents',
            'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']:
>>     data[x]=le.fit_transform(data[[x]])
>> data.head()
```



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Prope
0	LP001002	1	0	0	0	0	5849	0.0	146.412162	360.0	1	
1	LP001003	1	1	1	0	0	4583	1508.0	128.000000	360.0	1	
2	LP001005	1	1	0	0	1	3000	0.0	66.000000	360.0	1	
3	LP001006	1	1	0	1	0	2583	2358.0	120.000000	360.0	1	
4	LP001008	1	0	0	0	0	6000	0.0	141.000000	360.0	1	

Fig

The label encoder encodes all the object and categorical values into unit numbers.

Which in turn helps the machine learning model to fit these attributes correctly.

4.1.7 Outlier removal:

```
>> li=[]
>> for x in range(0, data.shape[0]-1):
>>     if data.iloc[x, 6]>30000:
>>         li.append(x)
>>     elif data.iloc[x, 7]>30000:
```

```
>> li.append(x)

>>data.drop(li,axis=0,inplace=True)
```

The above code will remove the datasets which have applicant income and co applicant income greater than 30,000. As they all are outliers of our dataset they can bias or can harm the accuracy of our machine learning model.

4.1.8Applying the machine learning model

```
>>from sklearn.preprocessing import train_test_split

>>from sklearn.linear_model import LogisticRegression

>>from sklearn.ensemble import RandomForestClassifier

>>from sklearn.neighbors import KNeighborsClassifier

>>from sklearn.model_selection import GridSearchCV

>>from sklearn.preprocessing import StandardScaler

>>x=data.drop(labels=['Loan_Status','Loan_ID'],axis=1)

>>y=data.Loan_Status
```

4.1.8.1 splitting the dataset into training and testing dataset

```
>>xtr,xte,ytr,yte=train_test_split(x,y,test_size=0.27,random_state=42)
```

4.1.8.2 Scaling the datasets

```
>>sc_X=StandardScaler()

>>xtr=sc_X.fit_transform(xtr)

>>xte= sc_X.transform(xte)
```

4.1.9 K neighbor classifier with hyperparameter tuning using grid search

```
>>k_range = list(range(1,31))

>>weight_options = ["uniform", "distance"]

>>param_grid = dict(n_neighbors = k_range, weights = weight_options)

>>grid=GridSearchCV(estimator=KNeighborsClassifier(),param_grid=param_grid,cv = 10, scoring = 'accuracy')

>>grid.fit(x,y)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py:814: D
DeprecationWarning)
GridSearchCV(cv=10, error_score='raise-deprecating',
            estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
            metric='minkowski',
            metric_params=None, n_jobs=None,
            n_neighbors=5, p=2,
            weights='uniform'),
            iid='warn', n_jobs=None,
            param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
            13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
            23, 24, 25, 26, 27, 28, 29, 30],
            'weights': ['uniform', 'distance']},
            pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
            scoring='accuracy', verbose=0)
```

Fig

4.1.9.1 measuring the accuracy and errors of trained machine learning model

```
>>print (grid.best_score_)

>>print (grid.best_params_)

>>print (grid.best_estimator_)
```

```
0.6909090909090909
{'n_neighbors': 26, 'weights': 'uniform'}
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=26, p=2,
                    weights='uniform')
```

Fig

4.1.10 Random forest classifier with hyperparameter tuning using grid search

```
>>randf=RandomForestClassifier()

>>pparam_grid = dict(n_estimators=list(range(10,200,10)),max_depth=list(range(2,10)))
```

```
>>ggrid=GridSearchCV(estimator=RandomForestClassifier(),param_grid=pparam_grid,cv = 10, scoring
= 'accuracy')
```

4.1.10.1 measuring the accuracy and errors of trained machine learning model

```
>>print (ggrid.best_score_)
```

```
>>print (ggrid.best_params_)
```

```
>>print (ggrid.best_estimator_)
```

```
0.8132231404958677
{'max_depth': 2, 'n_estimators': 50}
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=2, max_features='auto', max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=50,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)
```

Fig

4.1.11 Logistic regression with hyperparameter tuning

```
>>lr=LogisticRegression()
```

```
>>lr.fit(xtr,ytr)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432:
FutureWarning)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='warn', n_jobs=None, penalty='l2',
                    random_state=None, solver='warn', tol=0.0001, verbose=0,
                    warm_start=False)
```

Fig

```
>>ypr=lr.predict(xte)
```

4.1.11.1measuring the accuracy and errors of trained machine learning model

```
>>metrics.accuracy_score(yte,ypr)
```

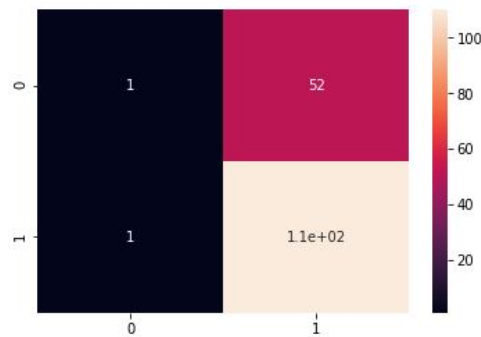
```
0.7771084337349398
```

4.1.12 Comparison and evaluation all three machine learning models

confusion matrix of knn:


```
>>knn=KNeighborsClassifier(algorithm='auto',leaf_size=30,metric='minkowski',
metric_params=None,n_jobs=None,n_neighbors=26,p=2,weights='uniform')
>>knn.fit(xtr,ytr)
```

```
>>fpr, tpr, threshold = metrics.roc_curve(yte,p)
>>sns.heatmap(metrics.confusion_matrix(yte,p),annot=True)
```

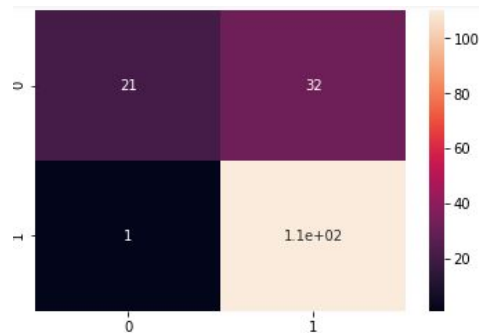


Fig

Confusion matrix of logistic regression

```
>>lr=LogisticRegression()
>>lr.fit(xtr,ytr)
```

```
>>ypr=lr.predict(xte)
>>sns.heatmap(metrics.confusion_matrix(yte,ypr),annot=True)
```



Fig

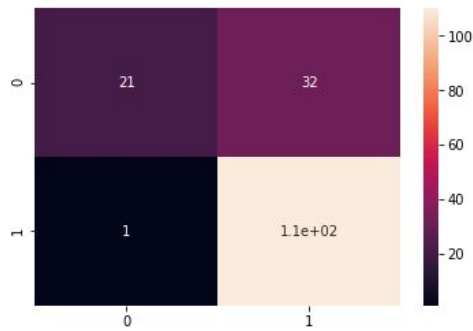
Confusion matrix of random forest :

```
>>randf=RandomForestClassifier(bootstrap=True,class_weight=None,criterion=
'gini', max_depth=2, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
```

```

min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=50,
n_jobs=None, oob_score=False, random_state=None,
verbose=0, warm_start=False)
>>randf.fit(xtr,ytr)
>>pp=randf.predict(xte)
>>sns.heatmap(metrics.confusion_matrix(yte,pp),annot=True)

```



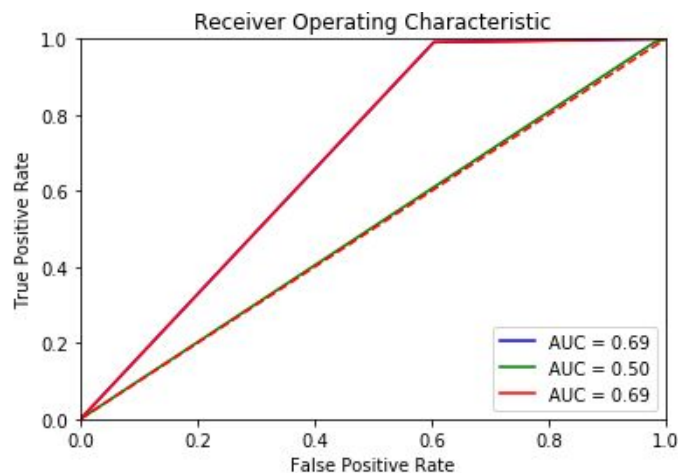
Fig

Roc curve :

```

>>fpr, tpr, threshold = metrics.roc_curve(yte,pp)
>>fpr1, tpr1, threshold1 = metrics.roc_curve(yte,p)
>>fpr2, tpr2, threshold2 = metrics.roc_curve(yte,ypr)
>>roc_auc = metrics.auc(fpr, tpr)
>>roc_auc1 = metrics.auc(fpr1, tpr1)
>>roc_auc2 = metrics.auc(fpr2, tpr2)
>>plt.title('Receiver Operating Characteristic')
>>plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
>>plt.plot(fpr1, tpr1, 'g', label = 'AUC = %0.2f' % roc_auc1)
>>plt.plot(fpr2, tpr2, 'r', label = 'AUC = %0.2f' % roc_auc2)
>>plt.legend(loc = 'lower right')
>>plt.plot([0, 1], [0, 1], 'r--')
>>plt.xlim([0, 1])
>>plt.ylim([0, 1])
>>plt.ylabel('True Positive Rate')
>>plt.xlabel('False Positive Rate')
>>plt.show()

```



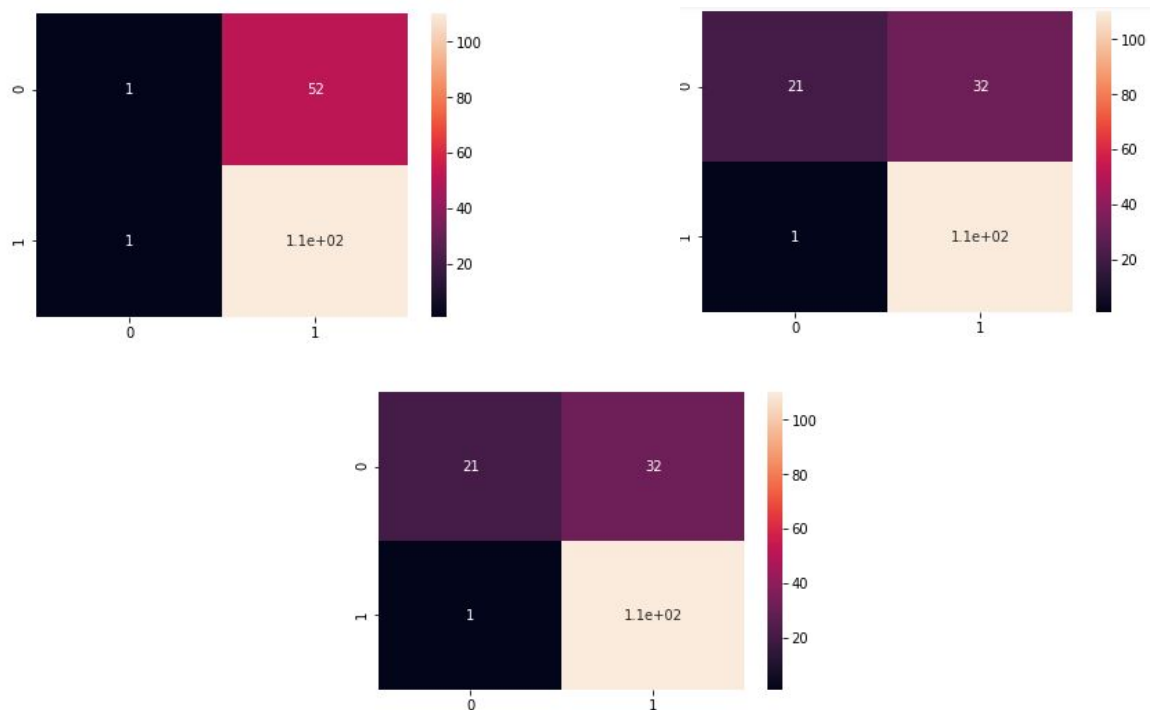
CHAPTER V

CONCLUSION

In conclusion we will see comparison of the models and tell which model came out to be more successful and satisfactory to our requirement and the requirement of our project.

Analysis of confusion matrix

For our Loan status prediction project, False Negatives Rate is the best metric to evaluate the model. Lower the number of false negatives, better the model is. In this project, False negative is when model predicting “an applicant will not get approved for a loan even though he will “. Our model cannot afford having higher False Negatives as it leads to negative impact on the investors and the credibility of the company. So, we evaluated our models using the number of False negatives and accuracies. The false negative values are mentioned in bottom left of each confusion matrix where 1 means approved



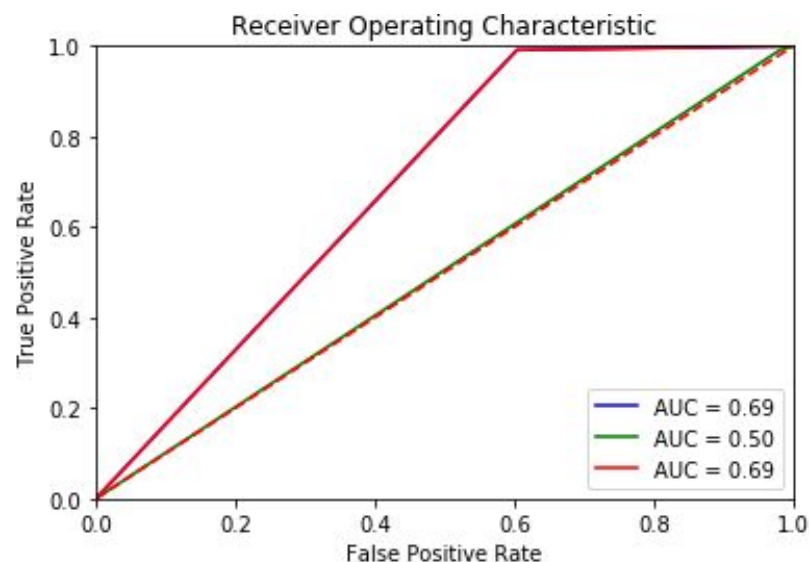
and 0 means not approved

Fig

by above given confusion matrices we can say that there is very slight difference between the algorithms so we can move on to our next evaluation expect

Analysis of roc curve

In a Receiver Operating Characteristic (ROC) curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test. This all can conclude to the point that whichever the model having a roc curve that passes through the upper left corner is more accurate than the one whose roc curve does not and if the curves of the algorithm passes with very slight difference then the accuracy can be understood by the means of the auc(area under the curve) more the area more accurate the algorithm. Now as you can see that the fig shows that there is a no difference between the roc curve of logistic regression and random forest, but there is a large difference between the knn's roc curve which is flat therefor telling us that this model is not suitable for our project so we can now take random forest and logistic regression in consideration .so moving on to our



next aspect of evaluation of algorithms.

Fig

Accuracy score

K- nearest neighbor classifier - 0.6909090909090909

Random forest classifier - 0.8132231404958677

Logistic regression classifier - 0.7987804878048781

Accuracy score is a metrics that is used to evaluate the trained model on the basis of its predictions on the test set data .this metrics provide a score to your machine learning model based on the percentage of the correct prediction your trained model can predict. Now by above mentioned accuracy score we can tell that there is very small difference mere 2% (round off value) and there is a difference of 12 % (round off value). So we can say that random forest is the best suited algorithm for these type of problem and these type of projects. and also if we look into the statistical terms logistic regression only take the input and apply the logistic/ sigmoid function on it and thereby giving the predictions, whereas in random forest we can say that the prediction can be decided by the majority prediction of the decision trees according to their forest size as in our project the size of the forest was 50 so this explains that there will be 50 decisions and the final prediction will be dependent on the majority of the similar decisions.

Result

The random forest classifier came out to be the most successful machine learning model on the basis of our evaluation tests (roc curve comparison, auc comparison, accuracy score) that we have done on the models.

CHAPTER VI

CITATIONS AND BIBLIOGRAPHY

Help from online sources

- <http://scikit-learn.org/stable/>.
- Few tutorials on cross validation from YouTube channel Data School
<https://www.youtube.com/user/dataschool>.
- <http://seaborn.pydata.org/>.
- Advanced algorithms like Boosting and Bagging from Machine Learning Mastery <https://machinelearningmastery.com/>
- For incites of notebooks and help in my project organisation
<https://www.kaggle.com/>
- For dataset <https://www.kaggle.com/>
- For working out my basic code for experimentation
<https://colab.research.google.com/>
- Very useful articles on various topics related to data science and data analytics
<https://towardsdatascience.com/>
- Statistical knowledge for almost every machine learning model
<https://www.coursera.org/learn/machine-learning> teacher - dr. Andrew ng

Citations from books

- For beginner level motivation and knowledge for machine learning models, data analysis and projects related to them – Hands-On Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems
- For introduction to python and how it can revolutionise the field of machine learning - Introduction to Machine Learning with Python: A Guide for Data Scientists