# 1.INTRODUCTION

In today's dynamic economic landscape, individuals and businesses often rely on loans to meet their financial goals and aspirations. The loan approval process, however, can be intricate and time-consuming, involving a thorough assessment of various factors to determine an applicant's creditworthiness. With the advent of advanced technologies and the increasing availability of data, the financial industry has witnessed a paradigm shift towards leveraging predictive analytics and machine learning to streamline and enhance the loan approval process.

The ability to accurately predict the likelihood of loan approval is crucial for financial institutions, as it not only optimizes decision-making but also ensures a fair and efficient allocation of resources. Traditional methods of assessing credit risk often involve manual scrutiny of extensive paperwork and historical data, which can be both time-intensive and prone to human error. In contrast, predictive models can analyse vast datasets, including financial histories, transaction patterns, and socio-economic indicators, to generate insights that facilitate more informed and consistent lending decisions.

This project aims to explore the realm of loan approval prediction by employing advanced machine learning algorithms and data analytics techniques. By harnessing the power of historical loan data, applicant information, and relevant financial indicators, we intend to develop a robust predictive model capable of assessing the probability of loan approval with a high degree of accuracy. This not only accelerates the decision-making process but also enhances the overall efficiency of financial institutions, enabling them to cater to the diverse needs of their clientele more effectively.

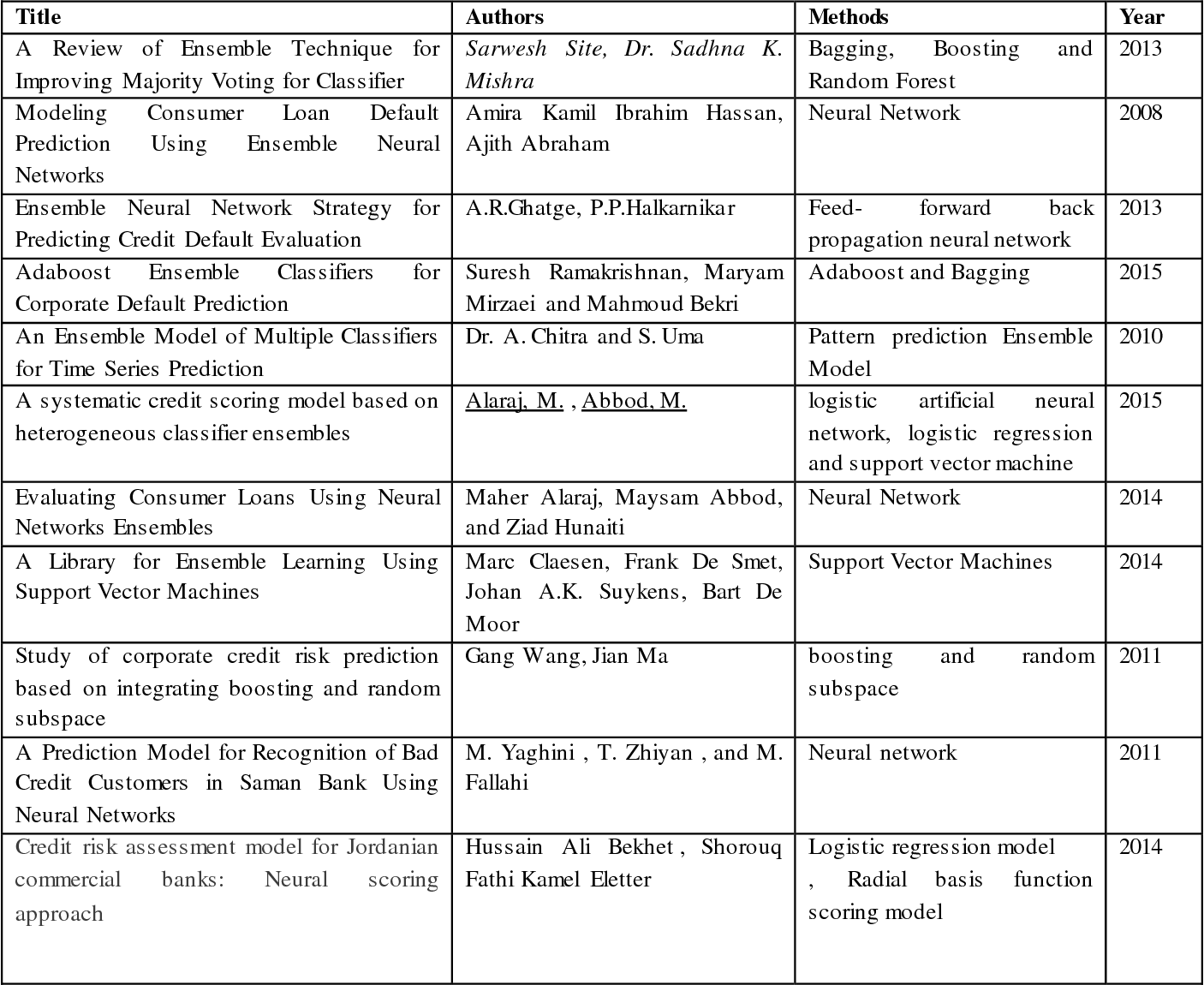
The significance of this endeavour lies in its potential to revolutionize the traditional lending landscape, making it more inclusive, transparent, and responsive to the evolving financial ecosystem. As we delve into the intricacies of loan approval prediction, we aim to contribute to the ongoing dialogue surrounding responsible and data-driven lending practices, ultimately fostering a more resilient and adaptive financial sector.

**2. LITERATURE SURVEY**

A recent development of machine learning techniques and data mining has led to an interest of implementing these techniques in various fields. The banking sector is no exclusion and the increasing requirements towards financial institutions to have robust risk management has led to an interest of developing current methods of risk estimation. Potentially, the implementation of machine learning techniques could lead to better quantification of the financial risks that banks are exposed to. Within the credit risk area, there has been a continuous development of the Basel accords, which provides frameworks for supervisory standards and risk management techniques as a guideline for banks to manage and quantify their risks. From Basel II, two approaches are presented for quantifying the minimum capital requirement such as the standardized approach and the internal ratings-based approach (IRB). There are different risk measures banks consider to estimate the potential loss they may carry in future. One of these measures is the expected loss (EL) a bank would carry in case of a defaulted customer. One of the components involved in ELestimation is the probability if a certain customer will default or not.

A prediction is a statement about what someone thinks will happen in the future. People make predictions all the time. Some are very serious and are based on scientific calculations, but many are just guesses. Prediction helps us in many things to guess what will happen after some time or after a year or after ten years.

Predictive analytics is a branch of advanced analytics that uses many techniques from data mining, statistics, modelling, machine learning, and artificial intelligence to analyse current data to make predictions. “Adyan Nur Alfiya tin, Hilman Taufiq and their friends work on the house price prediction. They use regression analysis and Particle Swarm Optimization (PSO) to predict house price”. One other similar work on the Mohamed El Moha dab, Belaid Bouikhalene and Said Safi to predict the rank for scientific research paper using supervised learning. Kumar Arun, Garg Ishan and Kaur Sanmeet work on bank loan prediction on how to bank approve a loan. They proposed a model with the help of SVM and Neural networks like machine learning algorithms. This literature review helps us carry out our work and propose a reliable bank loan prediction model. Manjeet et al (2018) there are seven types of variables that may influence consumer loan default; consumer’s annual income, debt-income ratio, occupation, home ownership, work duration and whether consumer possesses.



**3. SYSTEM ANALYSIS**

**1. Data Collection:**

Gather relevant data for training and testing the predictive model. This includes historical loan data, applicant information, credit scores, income details, employment history, and any other factors that may influence loan approval.

**2.Data Preprocessing:**

Clean and preprocess the data to handle missing values, outliers, and ensure consistency. Perform data normalization or standardization to bring all features to a similar scale. This step is crucial for the accuracy and effectiveness of the predictive model.

**3.Feature Engineering:**

Identify important features and consider creating new ones that may enhance predictive performance. This could involve transforming variables, creating interaction terms, or encoding categorical variables appropriately.

**4.Model Selection:**

Choose a suitable machine learning algorithm based on the characteristics of the data and the goals of the loan approval prediction system. Common algorithms for this task include decision trees, random forests, support vector machines, and gradient boosting.

**5.Model Training:**

Split the dataset into training and testing sets. Train the selected model using the training set and validate its performance on the testing set. Use techniques like cross-validation to ensure the model generalizes well to new data.

**6.Model Evaluation:**

Evaluate the model's performance using relevant metrics such as accuracy, precision, recall, and F1 score. Consider the business context and the importance of false positives and false negatives in the loan approval process.

**7.Hyperparameter Tuning:**

Fine-tune the model's hyperparameters to optimize its performance. This may involve using grid search or randomized search to find the best combination of hyperparameter values.

**8.Model Interpretability:**

Enhance the interpretability of the model, especially if it will be used in a regulated or customer-facing environment. Consider using techniques such as SHAP (SHapley Additive explanations) values to explain the model's predictions.

**9. Deployment:**

Once satisfied with the model's performance, deploy it to a production environment. This involves integrating the predictive model into the existing loan approval system. Consider containerization tools like Docker for easy deployment and management.

**10.Monitoring and Maintenance:**

Implement monitoring mechanisms to track the model's performance over time. Set up alerts for potential issues and regularly update the model to adapt to changing data patterns and ensure continued accuracy.

**11. Security and Compliance:**

Implement security measures to protect sensitive customer data. Ensure compliance with relevant regulations, such as GDPR or local data protection laws, to maintain the trust and privacy of customers.

**12.User Interface:**

Develop a user interface for stakeholders involved in the loan approval process. This interface could display relevant information and model predictions to support decision-making. - Fine-tune the model's hyperparameters to optimize its performance. This may involve using grid search or randomized search to find the best combination of hyperparameter values.

**PROCEDURE TO SOLVE THE GIVEN PROBLEM**

* Linear regression
* Decision tree
* Random forest

**3.1 LINEAR REGRESSION**

Linear regression is a supervised machine learning method that is used by the Train Using AutoML tool and finds a linear equation that best describes the correlation of the explanatory variables with the dependent variable. This is achieved by fitting a line to the data using least squares. The line tries to minimize the sum of the squares of the residuals. The residual is the distance between the line and the actual value of the explanatory variable. Finding the line of best fit is an iterative process. Advantages of linear regression algorithm:

• Linear regression performs exceptionally well for linearly separable data.

• Easier to implement, interpret and efficient to train.

• It handles overfitting well using dimensionally reduction techniques, regularization, and cross-validation.

• One more advantage is the extrapolation beyond a specific data set.



**3.2 DECISION TREE**

Decision trees are a nonparametric supervised learning method used for classification and regression. The deeper the tree, the more complex the decision rules and the fitter the model. Decision tree uses the tree representation to solve the problem. In which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. The primary challenge in the decision tree implementation is to identify the attributes. There are two popular attribute selection measures they are Entropy and Gini index. Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.



**3.3 RANDOM FOREST**

Random forest is one of the most popular algorithms in most use cases / projects across industries. Its fast, easier to implement, needs lesser data, doesn’t require extensive training and produces almost equally good results.

Again, because so many API libraries exist for several machine learning algorithms, the code for all simple machine learning algorithms is straightforward.



**4. IMPLEMENTATION**

**4.1 What is Python: -**

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers must type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

• Machine Learning

• Web frameworks like Django (used by YouTube, Instagram, Dropbox)

• Image processing (like OpenCV, Pillow)

• Web scraping (like Scrapy, Beautiful Soup, Selenium)

• Test frameworks

• Multimedia

**Advantages of Python:-**

Let’s see how Python dominates over other languages.

**1.Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.  **2.Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done. **5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. These further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class 14 allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages. Any doubts till now in the advantages of Python? Mention in the comment section.

**4.2 Advantages of Python Over Other Languages**

**1. Less Coding**

Almost all the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable** Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support. 15 The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac, or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis, and machine learning, automate things, do web scraping, and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1.Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing** **and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbon Nelle. The reason it is not so famous despite the existence of Bryton is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary. This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python:** -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later in the same Interview, Guido van Rossum continued: " I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

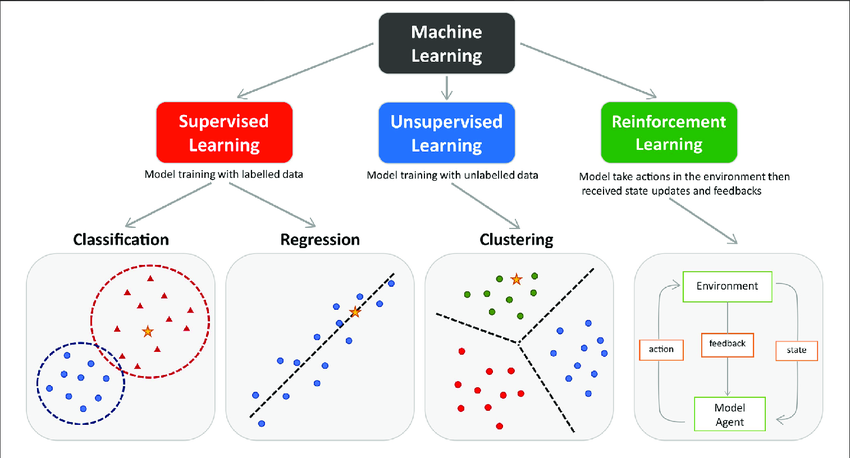
**4.3 What is Machine Learning**: -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is like the "learning" exhibited by the human brain understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

**Categories Of Machine Learning: -**

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.



Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section. Unsupervised learning involves modelling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

**Need for Machine Learning**

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

**Challenges in Machines Learning: -**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

**Quality of data −** Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task −** Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons −** As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems −** Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of overfitting & underfitting −** If the model is overfitting or underfitting, it cannot be represented well for the problem.

**Curse of dimensionality −** Another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment −** Complexity of the ML model makes it quite difficult to be deployed in real life.

**Applications of Machines Learning: -**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML.

• Emotion analysis

• Sentiment analysis

• Error detection and prevention

• Weather forecasting and prediction

• Stock market analysis and forecasting

• Speech synthesis

• Speech recognition

• Customer segmentation

• Object recognition

• Fraud detection

• Fraud prevention

• Recommendation of products to customer in online shopping

**How to Start Learning Machine Learning?**

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is the Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

**How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

1. **Learn Linear Algebra and Multivariate Calculus**

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important 21 as you will have to implement many ML algorithms from scratch.

1. **Learn Statistics**

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So, it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

1. **Learn Python**

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc. So, if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

**4.4 Learn Various ML Concepts**

Now that you are done with the prerequisites, you can move on to learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

1. **Terminologies of Machine Learning**

**• Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.

**• Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

• **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.

**• Training** – The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

**• Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

**(b) Types of Machine Learning**

**• Supervised Learning** – This involves learning from a training dataset with labelled data using classification and regression models. This learning process continues until the required level of performance is achieved.

**• Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data to learn more and more about the data itself using factor and cluster analysis models.

**• Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labelled data. Using labelled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.

**• Reinforcement** **Learning** – This involves learning optimal actions through trial and error. So, the next action is decided by learning behaviours that are based on the current state and that will maximize the reward in the future.

**Advantages of Machine learning: -**

1. **Easily identifies trends and patterns** –

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

1. **No human intervention needed (automation)**

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and improve the algorithms on their own. A common example of this is anti-virus software; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

1. **Continuous Improvement**

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data, you have keeps growing, your algorithms learn to make more accurate predictions faster.

1. **Handling multi-dimensional and multi-variety data**

Machine Learning algorithms are good at handling data that are multi-dimensional and multivariety, and they can do this in dynamic or uncertain environments.

1. **Wide Applications**

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Disadvantages of Machine Learning: -**

1. **Data Acquisition**

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

1. **Time and Resources**

ML needs enough time to let the algorithms learn and develop enough to fulfil their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

1. **Interpretation of Results**

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

1. **High error-susceptibility**

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

**Python Development Steps: -**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt. sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. These 25 releases included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

• Print is now a function.

• Views and iterators instead of lists

• The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.

• There is only one integer type left, i.e. int. long is int as well.

• The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.

• Text Vs. Data Instead of Unicode Vs. 8-bit

**Purpose: -**

We demonstrated that our approach enables successful segmentation of intra-retinal layers— even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type of system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional, and procedural, and has a large and comprehensive standard library.

• Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP. 26

• Python is Interactive − you can sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you must scan, read and/or understand to troubleshoot problems or tweak behaviours. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Modules Used in Project: -**

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

♣ A powerful N-dimensional array object

♣ Sophisticated (broadcasting) functions

♣ Tools for integrating C/C++ and Fortran code.

♣ Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to integrate with a wide variety of databases seamlessly and speedily.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyse. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn.**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**5.SOFTWARE REQUIREMENT SPECIFICATION**

**5.1 Requirements Specification:**

Requirement Specification provides a high secure storage to the web server efficiently. Software requirements deal with software and hardware resources that need to be installed on a serve which provides optimal functioning for the application. These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application.

* + 1. **HARDWARE REQUIREMENTS:**

**1.Processing Power:**

-For small to medium-sized datasets, a standard multi-core CPU can be sufficient.

-For larger datasets or complex models, consider using a high-performance CPU or distributed computing resources.

**2.Memory (RAM):**

- The amount of RAM required depends on the size of your dataset. For moderate datasets, 16GB to 32GB of RAM may be suitable.

- Larger datasets or more complex models may benefit from 64GB or more.

**3.Storage:**

- Adequate storage space is necessary for storing datasets, model files, and related resources.

- SSDs are preferable for faster data access and model training.

**4.Graphics Processing Unit (GPU):**

- GPUs can significantly speed up the training of deep learning models. Consider using a GPU, especially for neural network-based approaches.

- NVIDIA GPUs are commonly used for deep learning tasks.

**5.Network Connectivity:**

- A reliable internet connection may be required for downloading datasets, model libraries, or updates.

**5.1.2 SOFTWARE** **REQUIREMENTS**

**1. Operating System:**

- Choose an operating system compatible with your preferred machine learning libraries and frameworks. Linux distributions (e.g., Ubuntu) are often preferred for machine learning tasks.

**2.Python:**

- Python is widely used for machine learning. Install the latest version of Python and package management tools like pip.

3**.Development Environment:**

- Use an integrated development environment (IDE) such as Jupyter Notebooks, PyCharm, or VSCode for code development and experimentation.

**4. Machine Learning Libraries:**

- Install popular machine learning libraries like:

- Scikit-learn for traditional machine learning algorithms.

**5. Data Manipulation and Analysis:**

- Pandas for data manipulation and analysis.

- NumPy for numerical operations.

**6. Version Control:**

- Use version control systems like Git to manage code versions.

**7.Monitoring and Logging:**

- Implement tools for monitoring model performance and logging. This could include libraries like Tensor Board for deep learning models.

**5.2 FUNCTIONAL REQUIREMENTS:**

**1.User Authentication**

The system shall provide user authentication mechanisms to control access to the application. Users will be assigned roles with specific permissions.

**2.Data Ingestion**

The system shall allow the ingestion of historical loan data and applicant information from external sources. The data should include relevant features such as credit scores, income details, and employment history.

**3.Data Preprocessing**

The system shall perform data preprocessing tasks, including handling missing values, outlier detection, and normalization or standardization of data.

**4.Feature Engineering**

The system shall identify and engineer relevant features to enhance the predictive model's performance. Feature engineering may involve transforming variables or creating new features.

**5.Model Training and Testing**

The system shall split the dataset into training and testing sets and train the predictive model using machine learning algorithms. Model performance shall be evaluated through testing, and cross-validation techniques shall be employed.

**6.Real-Time Prediction**

If required, the system shall support real-time loan approval predictions. It should have endpoints for receiving and responding to real-time prediction requests.

**7.Model Interpretability**

The system shall provide tools for interpreting and explaining the predictions made by the model, promoting transparency, and understanding.

**8.User Interface**

Optionally, the system may include a user interface to display relevant information and predictions for stakeholders involved in the loan approval process.

**5.3. NON-FUNCTIONAL REQUIREMENTS**

**1.Performance**

The system shall be capable of handling large datasets and provide predictions within an acceptable response time, even during peak usage.

**2.Security**

The system shall implement robust security measures to protect sensitive customer data and ensure compliance with relevant data protection regulations.

**3.Scalability**

The system shall be designed to scale horizontally to accommodate growing data and user loads.

**4.Maintenance**

The system shall include mechanisms for monitoring model performance and facilitating regular updates to adapt to changing data patterns.

**5.4 PERFORMANCE REQUIREMENTS**

Performance requirements for a Loan Approval Prediction System are crucial to ensure that the application can handle the expected workload efficiently. Below are key performance requirements:

**1. Response Time:**

**Requirement:** The system should provide loan approval predictions within [insert specific time] seconds for both batch processing and real-time requests.

**Rationale:** Faster response times improve user satisfaction and operational efficiency, allowing for quicker decision-making in the loan approval process.

**2.Scalability:**

**Requirement:** The system should scale horizontally to handle an increase in the number of users and the volume of loan applications.

**Rationale:** Scalability ensures that the system can accommodate growing demand without sacrificing performance.

**3. Throughput:**

**Requirement:** The system should be able to process a minimum of [insert specific number] loan applications per hour.

**Rationale:** A high throughput is essential to meet the demands of a large number of loan applicants and maintain efficient processing.

**4. Concurrency:**

**Requirement:** The system should support at least [insert specific number] concurrent users making loan approval requests.

**Rationale:** Concurrent user support is crucial to handle multiple users accessing the system simultaneously without degradation in performance.

**5. Availability:**

**Requirement:** The system should have an availability of at least [insert specific percentage], ensuring it is operational during business hours.

**Rationale:** High availability minimizes downtime and ensures that the loan approval process is consistently accessible to users.

**6. Reliability:**

**Requirement:** The system should have a reliability rate of at least [insert specific percentage], minimizing the occurrence of errors or system failures.

**Rationale:** Reliable systems contribute to the overall trustworthiness of the loan approval process.

**7. Resource Utilization:**

**Requirement:** The system should utilize hardware resources, such as CPU and memory, efficiently, with a target utilization rate of [insert specific percentage].

**Rationale:** Efficient resource utilization contributes to overall system stability and performance.

**8. Data Processing Speed:**

**Requirement:** The system should process large datasets for training the predictive model within [insert specific time].

**Rationale:** Efficient data processing is crucial during the training phase to ensure timely model updates and improvements.

**9. Real-Time Predictions:**

**Requirement:** Real-time predictions should be generated within [insert specific time] seconds of receiving a request.

**Rationale:** Real-time predictions support timely decision-making in scenarios where immediate responses are required.

**10. Load Testing:**

**Requirement:** The system should undergo regular load testing to ensure that it can handle peak loads without significant degradation in performance.

**Rationale:** Load testing helps identify performance bottlenecks and ensures that the system can handle variations in workload.

**11. Monitoring:**

**Requirement:** Implement comprehensive monitoring tools to track system performance, detect anomalies, and generate alerts for potential issues.

**Rationale:** Monitoring is essential for proactive identification and resolution of performance-related issues.

**12. Network Latency:**

**Requirement:** The system should be designed to minimize network latency, aiming for response times that are unaffected by network delays.

**Rationale:** Reduced network latency improves the overall responsiveness of the system.

**5.5 FEASIBILITY STUDY**

A feasibility study for loan approval prediction involves assessing the viability and potential success of implementing a predictive model to evaluate loan applications. This study helps in understanding the technical, economic, operational, and legal aspects of deploying a loan approval prediction system. Here's a step-by-step guide to conduct a feasibility study:

**1. Define Objectives:**

Clearly outline the objectives of implementing a loan approval prediction model. Identify the specific goals, such as improving efficiency, reducing risks, or enhancing customer satisfaction.

**2. Scope of the Study:**

Define the scope of the study, including the types of loans (personal, business, mortgage) and the target customer segments.

**3. Data Collection and Analysis:**

Gather historical data on approved and rejected loan applications. Analyze the data to identify patterns and trends that can be used to train a predictive model. Key data may include income, credit score, employment history, debt-to-income ratio, and other relevant factors.

**4. Regulatory Compliance:**

Investigate and ensure that the implementation of a loan approval prediction model complies with local and international regulations. Address any legal or ethical considerations related to data privacy and fairness.

**5. Technology Infrastructure:**

Assess the existing technology infrastructure to determine if it can support the implementation of a predictive model. Consider factors such as data storage, processing power, and integration with existing systems.

**6. Cost-Benefit Analysis:**

Conduct a thorough cost-benefit analysis to estimate the financial implications of implementing the loan approval prediction system. Include costs related to technology, personnel, training, and any potential benefits such as increased approval rates or reduced default rates.

**7. Risk Analysis:**

Identify potential risks associated with implementing a loan approval prediction model. Consider risks related to data security, model accuracy, regulatory compliance, and customer acceptance.

**8. Technical Feasibility:**

Evaluate the technical feasibility of developing and maintaining the predictive model. Assess the availability of necessary skills, tools, and technologies. Consider whether the model can be integrated with existing systems.

**9. Operational Feasibility:**

Determine the impact of the loan approval prediction system on daily operations. Assess whether staff members can adapt to the new system and whether any additional training is required.

**10. Timeline:**

Develop a timeline for the implementation of the loan approval prediction model. Include milestones for data preparation, model development, testing, and deployment.

**11. Stakeholder Communication:**

Communicate with stakeholders, including management, staff, and customers, to gather feedback and address concerns. Ensure that there is support for the implementation of the predictive model.

**12. Conclusion and Recommendations:**

Summarize the findings of the feasibility study and provide recommendations on whether to proceed with the implementation of the loan approval prediction system.

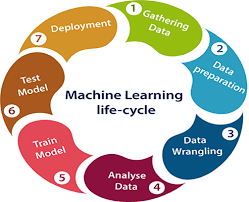
Once the feasibility study is complete, stakeholders can use the information to make informed decisions about whether to proceed with the development and implementation of a loan approval prediction model.

**6. METHODOLOGY**

**MACHINE LEARNING LIFE CYCLE:**

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps, which are given below:



* Gathering Data
* Data preparation
* Data Wrangling
* Analyse Data
* Train the model
* Test the model.
* Deployment

The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

**1. GATHERING DATA:**

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files**, **database**, **internet**, or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

This step includes the below tasks:

* Identify various data sources.
* Collect data.
* Integrate the data obtained from different sources.

**2. Data preparation:**

By performing the above task, we get a coherent set of data, also called as a **dataset**. It will be used in further steps.

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

* **Data exploration:**  
  It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.  
  A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.
* **Data pre-processing:**  
  Now the next step is preprocessing of data for its analysis.

**3. Data Wrangling:**

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

* Missing Values
* Duplicate data
* Invalid data
* Noise

So, we use various filtering techniques to clean the data. It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

**4. Data Analysis:**

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

* Selection of analytical techniques
* Building models
* Review the result.

The aim of this step is to build a machine learning model to analyse the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification, Regression, Cluster analysis, Association, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

**5. Train Model:**

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and features.

**6. Test Model:**

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

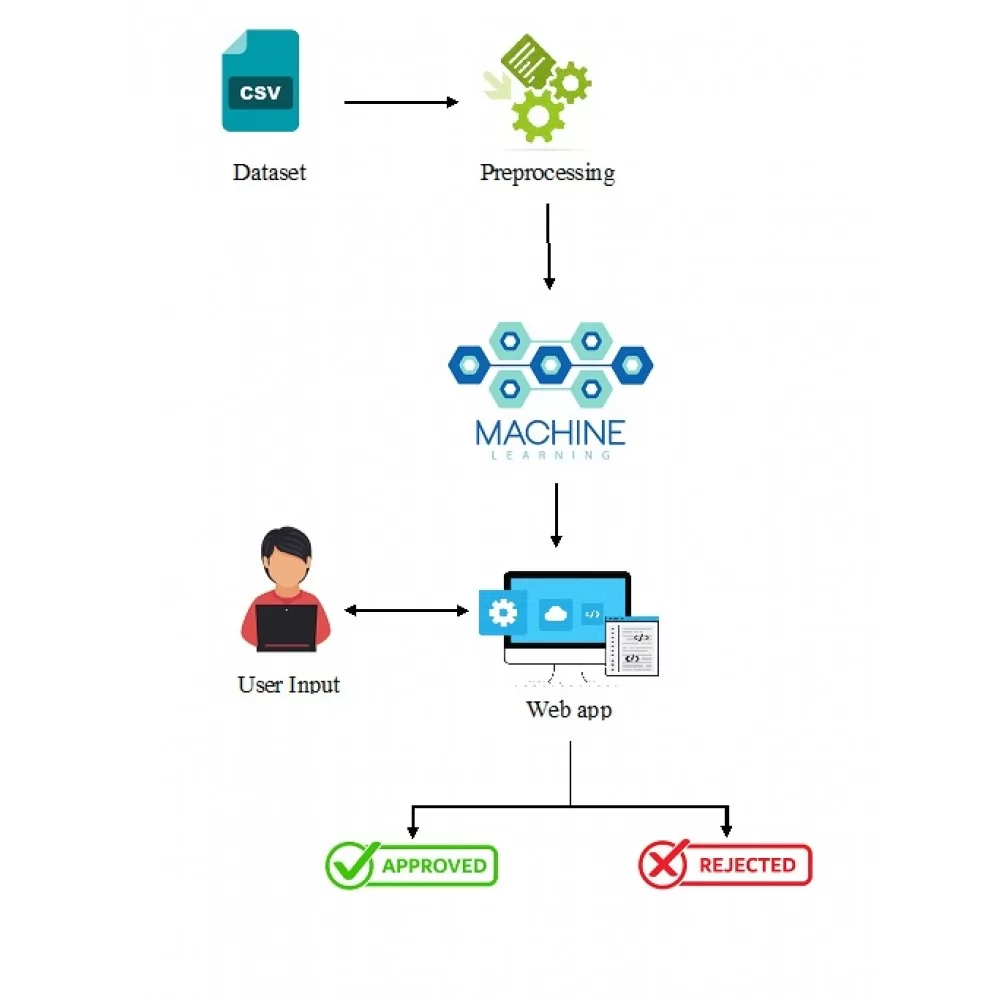
Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

**7. Deployment:**

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

**7.1ARCHITECTURE**



**7.2 UML DIAGRAMS**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Metamodel and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.

2. Provide extendibility and specialization mechanisms to extend the core concepts.

3. Be independent of programming languages and development process.

4. Provide a formal basis for understanding the modelling language.

5. Encourage the growth of OO tools market.

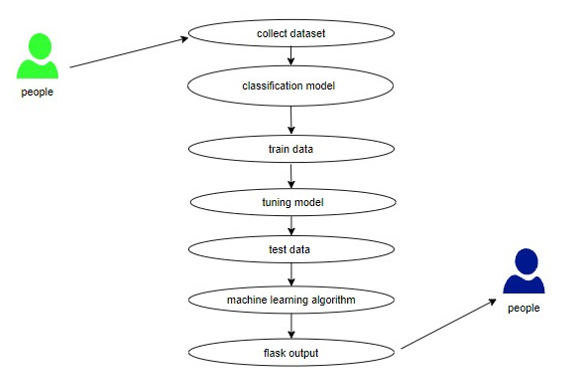
6. Support higher level development concepts such as collaborations, frameworks, patterns, and components.

7. Integrate best practices.

**7.3 USE CASE DIAGRAM**

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

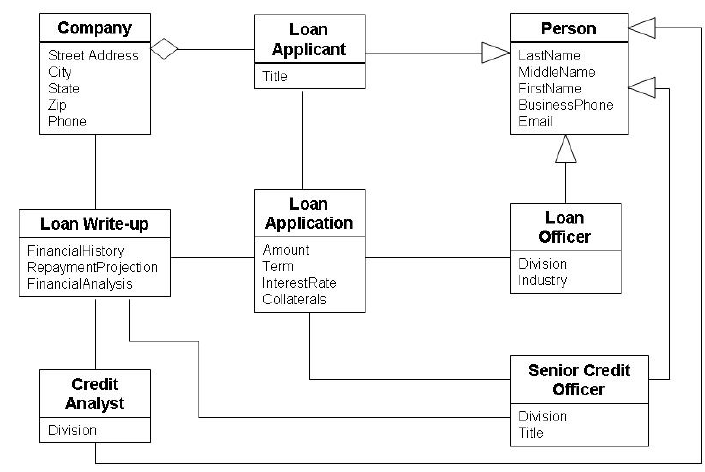
Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analysed the functionalities are captured in use cases. So, it can say that uses cases are nothing, but the system functionalities written in an organized manner.



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

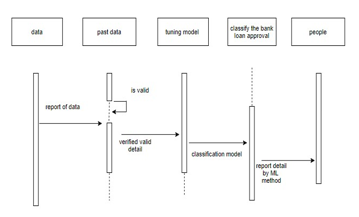
Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.



**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

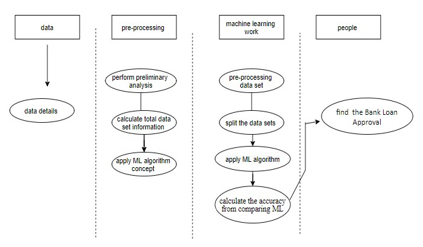
Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.



**ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams look like a flow chart, but it is not. It shows different flow like parallel, branched, concurrent and single.



**8. SOURCE CODE**

import pandas as pd.

import numpy as np.

import matplotlib as plt

import seaborn as sns

df = pd. read\_csv('loan\_approval\_dataset.csv')

df. head ()

df. Shape

df. drop (columns='loan\_id’, inplace=True)

df. isnull (). sum ()

df. dtypes

df ['Movable assets'] = df [' bank\_asset\_value'] + df [' luxury\_assets\_value']

#Immovable Assets

df['Immovable\_assets'] = df [' residential\_assets\_value'] + df [' commercial\_assets\_value']

df.drop(columns= [' bank\_asset\_value',' luxury\_assets\_value',' residential\_assets\_value',' commercial\_assets\_value'], inplace=True)

df. describe ()

df. head ()

sns. countplot (x = ' no\_of\_dependents', data = df). set\_title ('Number of Dependents')

fig, ax = plt. pyplot. subplots (1,2, figsize= (10, 5))

sns. Boxplot (x = ' education', y = ' income\_annum', data = df, ax=ax [0])

sns. violinplot (x = ' education', y = ' income\_annum', data = df, ax=ax [1])

sns. countplot (x=' self-employed', data = df, hue = ' education'). set\_title('Self-employed')

sns. lineplot (x = ' loan\_term', y = ' loan\_amount', data = df). set\_title ('Loan Amount')

sns. histplot (df [' cibil\_score'], bins = 30, kde = True, color = 'red')

fig, ax = plt. pyplot. subplots (1,2, figsize= (10,5))

sns. histplot (df ['Movable assets'], ax=ax [0], color='red')

sns. histplot (df ['Immovable assets'], ax=ax [1], color='blue')

sns. countplot (x = ' no\_of\_dependents', data = df, hue = ' loan status')

sns. countplot (x = ' education', hue = ' loan\_status', data = df). set\_title('Loan\_status')

sns. violinplot (x=' loan\_status', y=' income\_annum', data=df)

sns. lineplot (x=' loan\_term', y=' loan\_amount', data=df, hue=' loan\_status')

sns. violinplot (x=' loan\_status', y=' cibil\_score', data=df)

fig, ax = plt. pyplot. subplots (1,2, figsize= (10,5))

sns. histplot (x = 'Movable\_assets', data = df, ax=ax [0], hue = ' loan\_status')

sns. histplot (x = 'Immovable\_assets', data = df, ax=ax [1], hue = ' loan\_status')

# Label Encoding

df [' education'] = df [' education']. map ({' Not Graduate':0, ' Graduate':1})

df [' self-employed'] = df [' self-employed']. map ({' No':0, ' Yes':1})

df [' loan\_status'] = df [' loan\_status']. map ({' Rejected':0, ' Approved':1})

plt. pyplot. Figure (figsize= (10,10))

sns. Heatmap (df. corr (), annot = True, cmap='coolwarm')

fig, ax = plt. pyplot. subplots (1,2, figsize= (10, 5))

sns. Scatterplot (x='Movable\_assets', y = ' loan\_amount', data = df, ax=ax [0]). set

sns. Scatterplot (x='Immovable\_assets', y = ' loan\_amount', data = df, ax=ax [1]). set

sns. Scatterplot (x=' income\_annum', y = ' loan\_amount', data = df)

from sklearn. model selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split (df. drop (' loan\_status', axis=1), df[' loan\_status'], test\_size=0.2, random\_state=42)

from sklearn. Tree import DecisionTreeClassifier

# Create decision tree object

dtree = DecisionTreeClassifier ()

# Training the model using the training data

dtree.fit (X\_train, y\_train)

# Training Accuracy

dtree. Score (X\_train, y\_train)

# Predicting the Loan Approval Status

dtree\_pred = dtree. Predict(X\_test)

from sklearn. Ensemble import RandomForestClassifier.

# Create a random forest classifier

rfc = RandomForestClassifier ()

# Training the model using the training data

rfc. Fit (X\_train, y\_train)

# Training Accuracy

rfc. Score (X\_train, y\_train)

# Predicting the Loan Approval Status

rfc\_pred = rfc. predict(X\_test)

from sklearn. Metrics import confusion matrix.

fig, ax = plt. pyplot. subplots (1,2, figsize= (10,4))

sns. Heatmap (confusion matrix (y\_test, dtree\_pred), annot=True, ax=ax [0]). set\_title ('Decision Tree')

ax [0]. set\_xlabel('Predicted')

ax [0]. set\_ylabel('Actual')

sns. Heatmap (confusion matrix (y\_test, rfc\_pred), annot=True, ax=ax [1]). set\_title ('Random Forest')

ax [1]. set\_xlabel('Predicted')

ax [1]. set\_ylabel('Actual')

ax = sns. distplot (x = y\_test, hist = False, color = "r", label = "Actual Value")

sns. distplot (x = dtree\_pred, hist = False, color = "b", label = "Fitted Values", ax = ax)

plt. pyplot. Title ('Actual vs Fitted Values for Decision Tree Classifier')

ax = sns. distplot (x = y\_test, hist = False, color = "r", label = "Actual Value")

sns. distplot (x = rfc\_pred, hist = False, color = "b", label = "Fitted Values", ax = ax)

plt. Title ('Actual vs Fitted Values for Random Forest Classifier')

from sklearn. Metrics import classification report

print (classification report (y\_test, dtree\_pred))

print (classification report (y\_test, rfc\_pred))

from sklearn. Metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error

# Decision Tree Classifier

print ('R2 score: ', r2\_score (y\_test, dtree\_pred))

print ('Mean Squared Error: ', mean\_squared\_error (y\_test, dtree\_pred))

print ('Mean Absolute Error: ', mean\_absolute\_error (y\_test, dtree\_pred))

print('\n')

# Random Forest Classifier

print ('R2 score: ', r2\_score (y\_test, rfc\_pred))

print ('Mean Squared Error: ', mean\_squared\_error (y\_test, rfc\_pred))

print ('Mean Absolute Error: ', mean\_absolute\_error (y\_test, rfc\_pred))

**9. TESTING**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property function as a unit. The test data should be chosen such that it passed through all possible condition. The following is the description of the testing strategies, which were carried out during the testing period.

**9.1 SYSTEM TESTING**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to withstand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**9.2 MODULE TESTING**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time, and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately, and their corresponding results are obtained which reduces the process waiting time.

**9.3 INTEGRATION TESTING**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus, the mapping of jobs with resources is done correctly by the system.

**9.4 ACCEPTANCE TESTING**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

**9.5 TEST CASES:**

When creating test cases for a loan approval prediction system, it's essential to cover a variety of scenarios to ensure the system's accuracy, robustness, and compliance with requirements. Here are some test cases to consider:

**Positive Test Cases:**

**1. Standard Approval**:

- Input: Applicant with a good credit score, stable income, and a reasonable loan amount.

- Expected Output: Loan approval.

**2. Low-Risk Applicant:**

- Input: Applicant with an excellent credit score, high income, and a low loan amount.

- Expected Output: Loan approval.

**3. Joint Application:**

- Input: Married couple applying for a loan with combined income and good credit scores.

- Expected Output: Loan approval.

**4. Employment Stability:**

- Input: Applicant with a long employment history and stable income.

- Expected Output: Loan approval.

**5. Co-signer Scenario:**

- Input: Applicant with a lower credit score but a co-signer with a high credit score.

- Expected Output: Loan approval based on the co-signer's creditworthiness.

**Negative Test Cases:**

**6. Low Credit Score:**

- Input: Applicant with a low credit score.

- Expected Output: Loan rejection.

**7. Insufficient Income:**

- Input: Applicant with low income in relation to the loan amount.

- Expected Output: Loan rejection.

**8. Unstable Employment:**

- Input: Applicant with a history of frequent job changes.

- Expected Output: Loan rejection.

**9. High Debt-to-Income Ratio:**

- Input: Applicant with significant existing debts compared to income.

- Expected Output: Loan rejection.

**10. Applicant with Previous Loan Default:**

- Input: Applicant with a history of defaulting on previous loans.

- Expected Output: Loan rejection.

**Boundary Test Cases:**

**11. Minimum Credit Score:**

- Input: Applicant with the minimum acceptable credit score.

- Expected Output: Loan approval.

**12. Maximum Loan Amount:**

- Input: Applicant applying for the maximum allowed loan amount.

- Expected Output: Loan approval.

**13. Minimum Income:**

- Input: Applicant with the minimum acceptable income.

- Expected Output: Loan rejection.

**14. Maximum Debt-to-Income Ratio:**

- Input: Applicant with the maximum allowable debt-to-income ratio.

- Expected Output: Loan rejection.

**Edge Cases:**

**15. New Graduate:**

- Input: Recent graduate with no credit history.

- Expected Output: Evaluate based on other factors like education, employment prospects, etc.

**16. Self-Employed Applicant:**

- Input: Self-employed applicant with variable income.

- Expected Output: Loan approval/rejection based on stable income and financial history.

**17. Uncommon Employment Type:**

- Input: Applicant with an unusual or less common employment type.

- Expected Output: Evaluate based on the stability of income and employment.

**Compliance and Regulatory Test Cases:**

**18. Regulatory Compliance:**

- Verify that the system adheres to relevant lending regulations and policies.

**19. Data Privacy:**

- Ensure that the system handles personal and financial information securely and complies with data privacy regulations.

**20. Fair Lending Practices:**

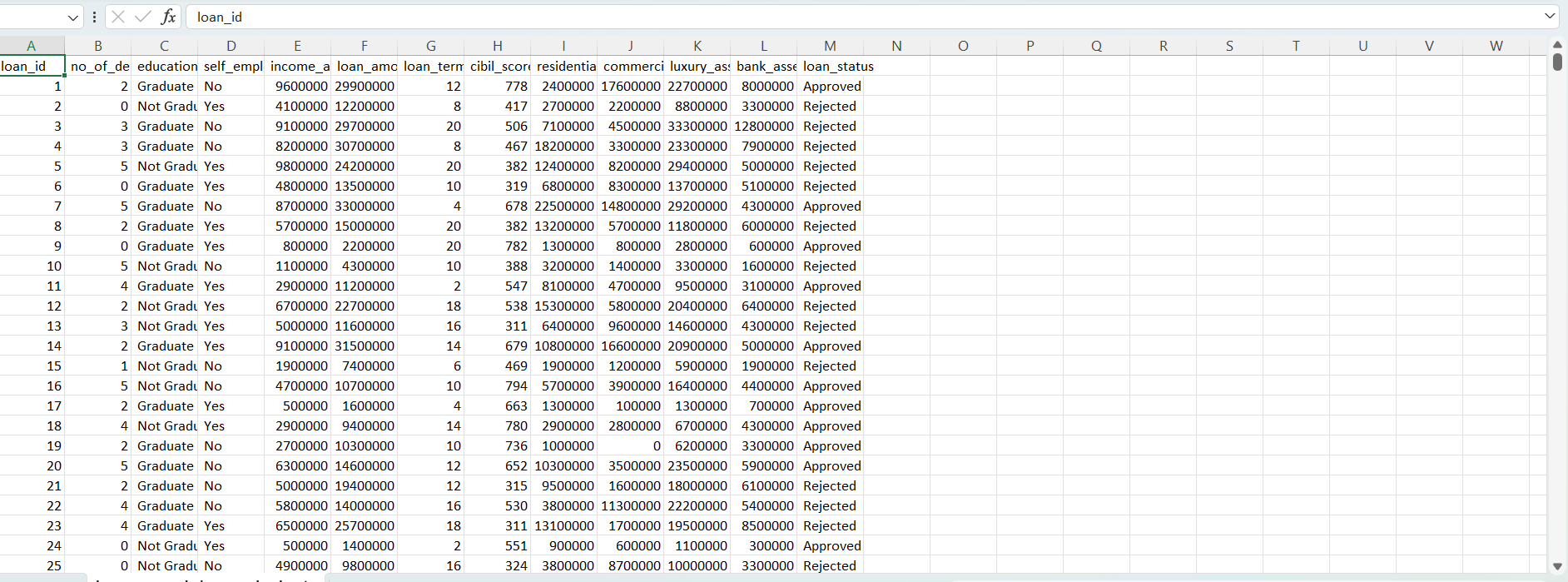
- Verify that the system does not discriminate based on factors such as race, gender, or other protected characteristics.

|  |  |  |  |
| --- | --- | --- | --- |
| Test CaseID | Scenario | Input Variables | Expected Output |
| TC01 | Standard Approval | Creditscore:750, Income: $80,000, LoanAmount: $50,000 | Loan Approval |
| TC02 | Low-Risk Applicant | Creditscore:800,  Income: $100,000,  Loan Amount: $20,000 | Loan Approval |
| TC03 | Joint Application | Applicant1: CreditScore:750,  Income: $60,000, <br> Applicant2:  CreditScore:760,  Income: $70,000 | Loan  Approval |
| TC04 | Employment Stability | CreditScore:720,  Income: $90,000,  LoanAmount: $40,000, Employment Years:10 | Loan Approval |
| TC05 | Co-signer Scenario | Applicant: CreditScore:650,  Income: $50,000,  <br>Co-signer: Credit Score:780 | Loan Approval |
| TC06 | Low Credit Score | CreditScore:500, Income: $60,000,  LoanAmount: $30,000 | Loan Rejection |
| TC07 | Insufficient Income | CreditScore:700,  Income:$30,000,  Loan amount:  $25,000 | Loan Rejection |
| TC08 | Unstable Employment | CreditScore:720,  Income:$80,000,  EmploymentYears:2 | Loan  Rejection |
| TC09 | High Debt-to-Income Ratio | CreditScore:600,  Income:$50,000  Existing Debt:$30,000 | Loan  Rejection |
| TC10 | Previous Loan Default | CreditScore:600,  Income:$70,000,  Loan Amount:$20,000,  Default History: Yes | Loan  Rejection |
| TC11 | Minimum Credit Score | CreditScore:300,  Income:$60,000,  Loan Amount:$10,000 | Loan  Rejection |
| TC12 | Maximum Loan Amount | CreditScore:780,  Income:$100,000,  Loan Amount:$500,000 | Loan  Rejection |
| TC13 | Minimum Income | CreditScore:700,  Income:$20,000,  Loan Amount:$15,000 | Loan  Rejection |
| TC14 | Maximum Debt-to-Income | CreditScore:720,  Income: $50,000,  Existing Debt:$25,000 | Loan  Rejection |
| TC15 | New Graduate | Credit Score : N/A,  Income: $40,000,  Loan Amount:$15,000,  Education: Recent  Graduate | Evaluate |
| TC16 | Self-Employed Applicant | CreditScore:750,  Income: Variable,  Loan Amount: $30,000 | Loan  Approved/Rejection |
| TC17 | Uncommon Employment Type | Credit Score:720,  Income: $60,000,  Employment Type: Freelancer | Loan  Approved/Rejection |
| TC18 | Regulatory Compliance | Validate adherence to lending regulation and policies | Compliant |
| TC19 | Data Privacy | Ensure secure handling of personal and financial information | Compliant |
| TC20 | Fair Lending Practices | Verify non-discrimination based on protected characteristics | Fair Evaluation |

**10. SCREENS**

The loan approval dataset is a collection of financial records and associated information used to determine the eligibility of individuals or organizations for obtaining loans from a lending institution. It includes various factors such as cibil score, income, employment status, loan term, loan amount, assets value, and loan status. This dataset is commonly used in machine learning and data analysis to develop models and algorithms that predict the likelihood of loan approval based on the given features.

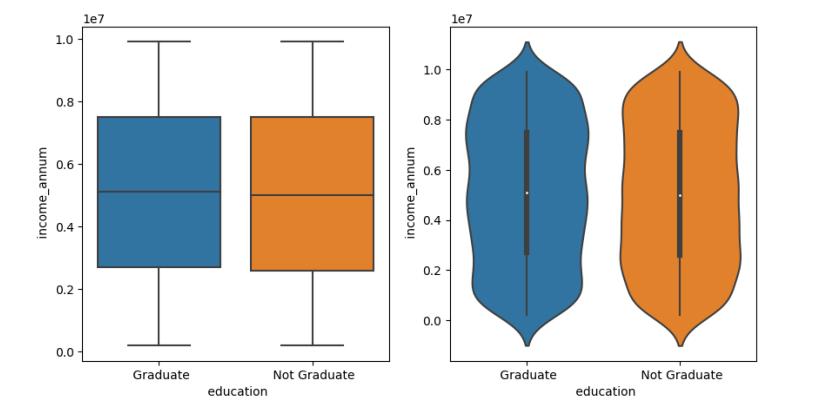
1. loan\_id
2. no\_of\_dependents
3. education
4. self-employed
5. income\_annum
6. loan\_amount
7. loan\_tenure
8. cibil\_score
9. residential\_asset\_value
10. commercial\_asset\_value
11. luxury\_asset\_value
12. bank\_assets\_value
13. loan\_status



**DATA VISULAZATION:**

A graph of different colored rectangular objects

Description automatically generated

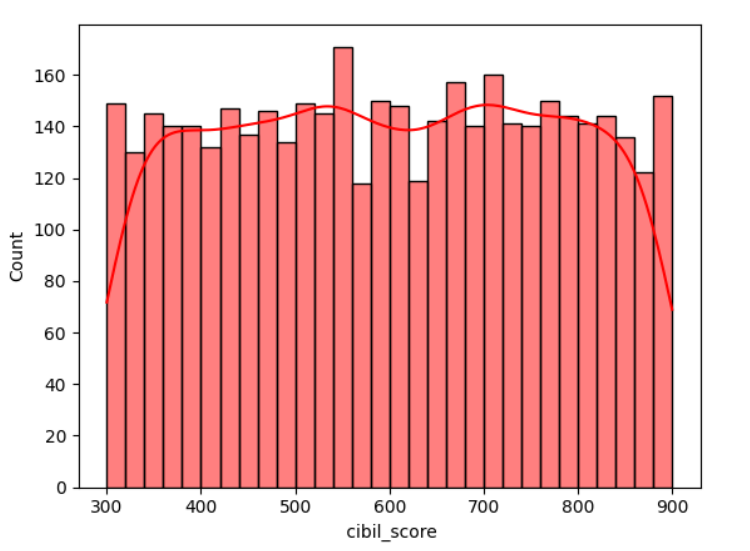


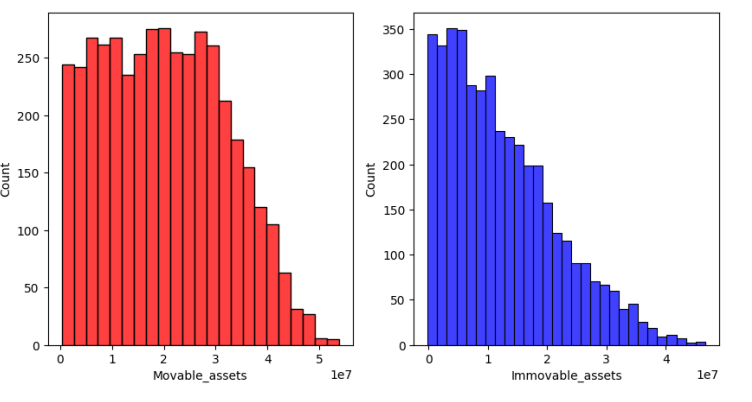
A graph of a self employed

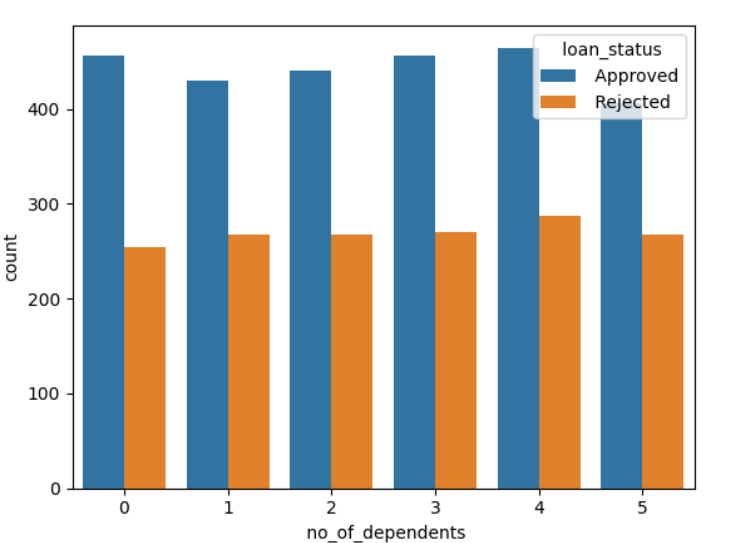
Description automatically generated

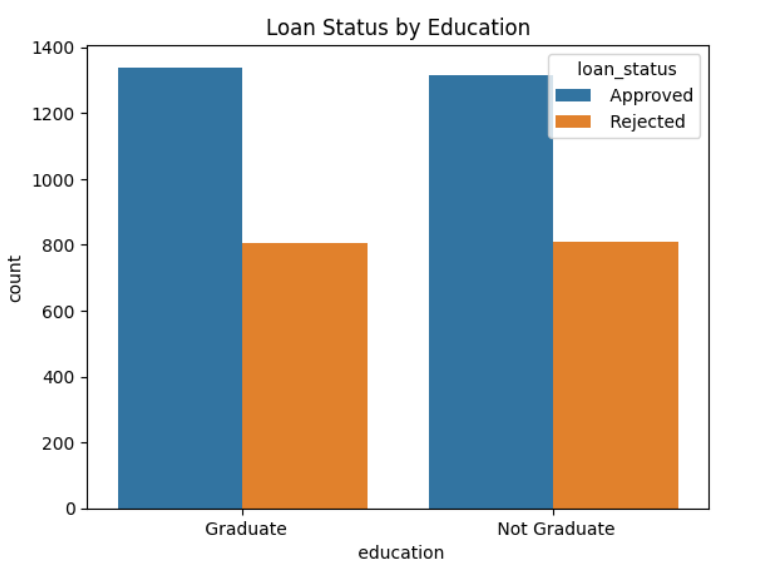
A graph showing a line of a graph

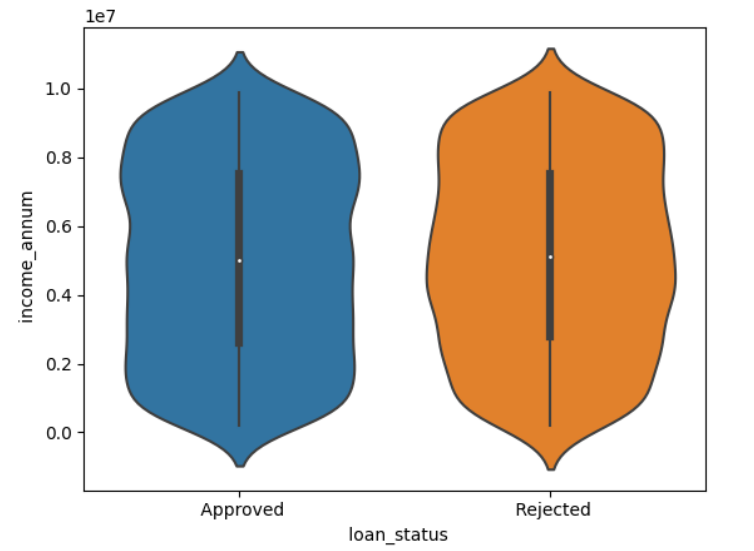
Description automatically generated with medium confidence

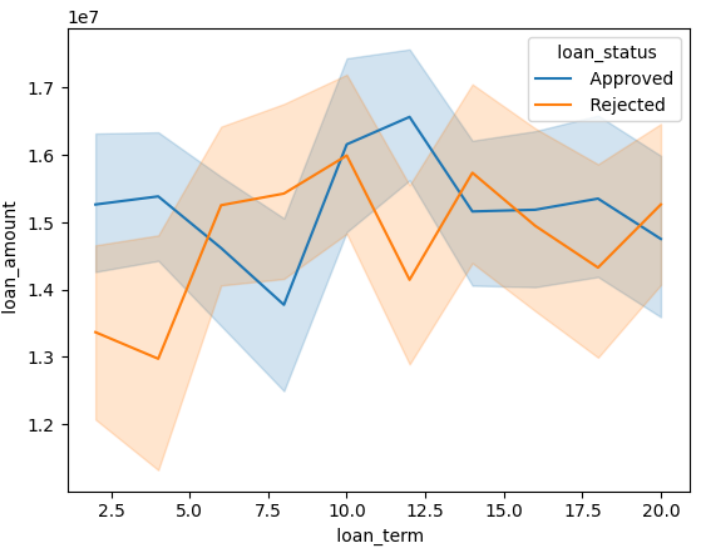


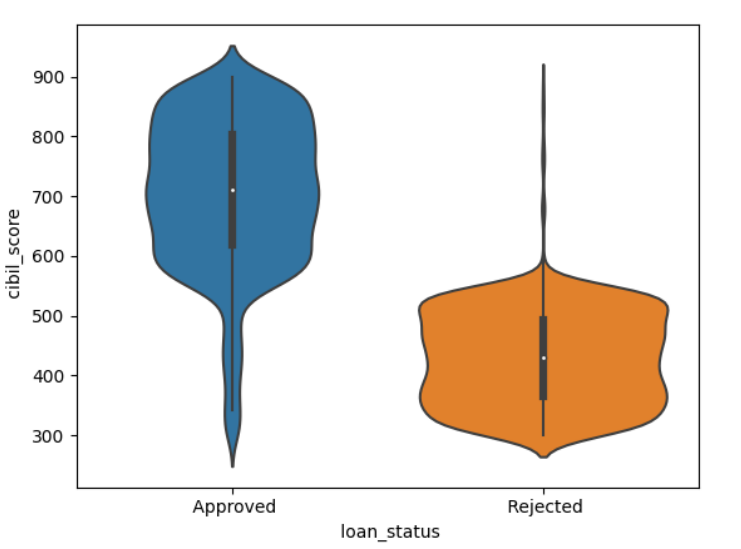


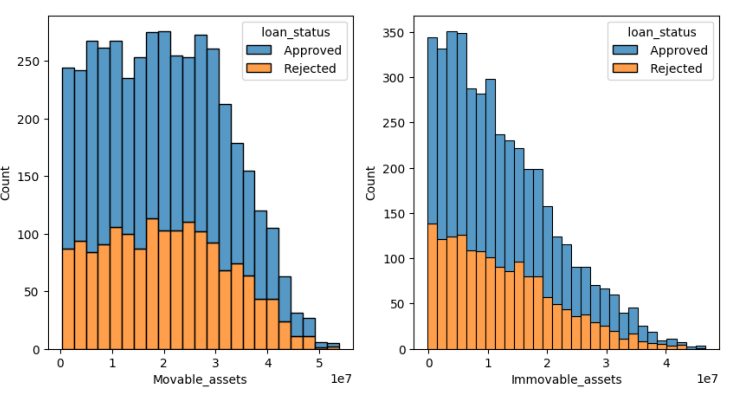


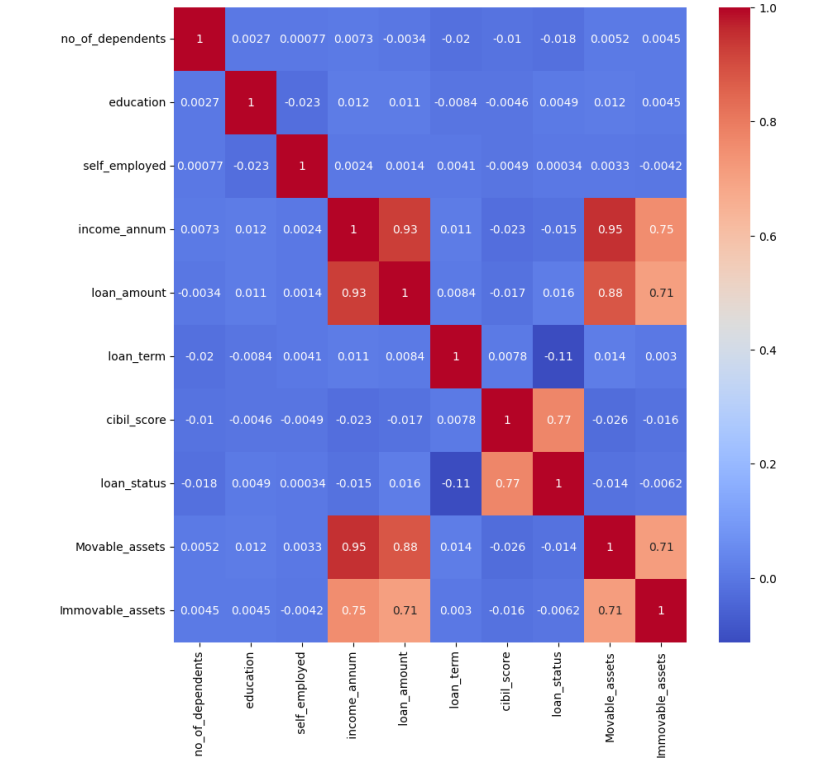


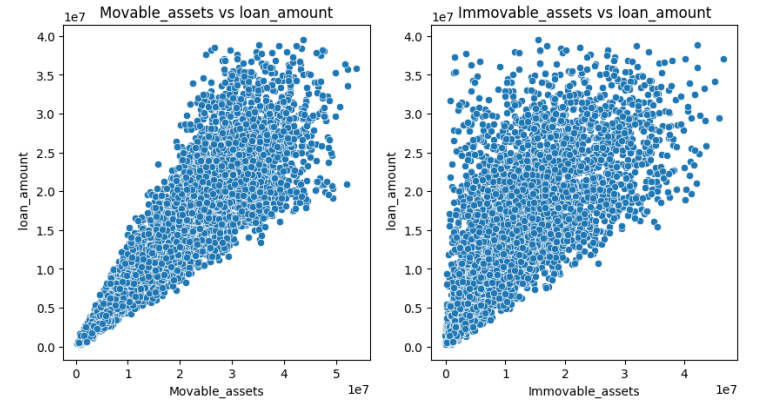


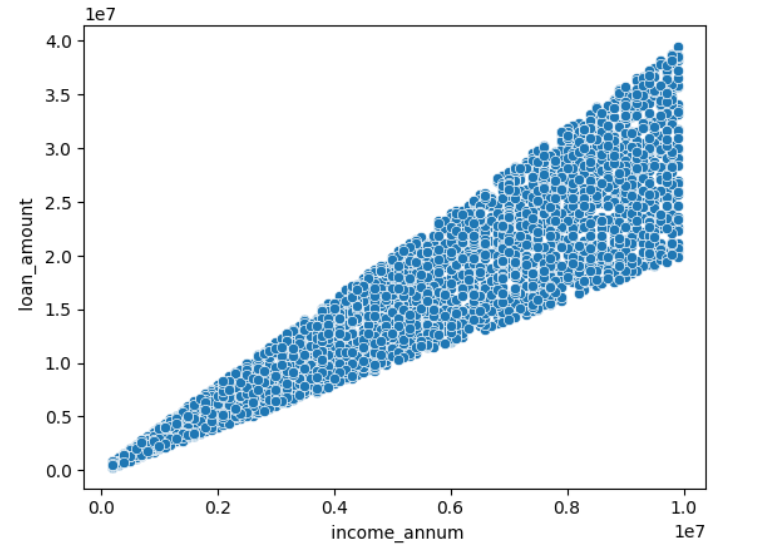






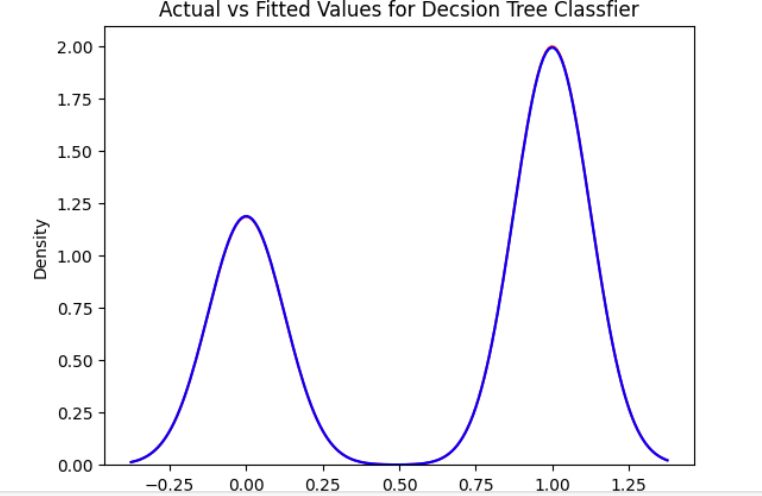


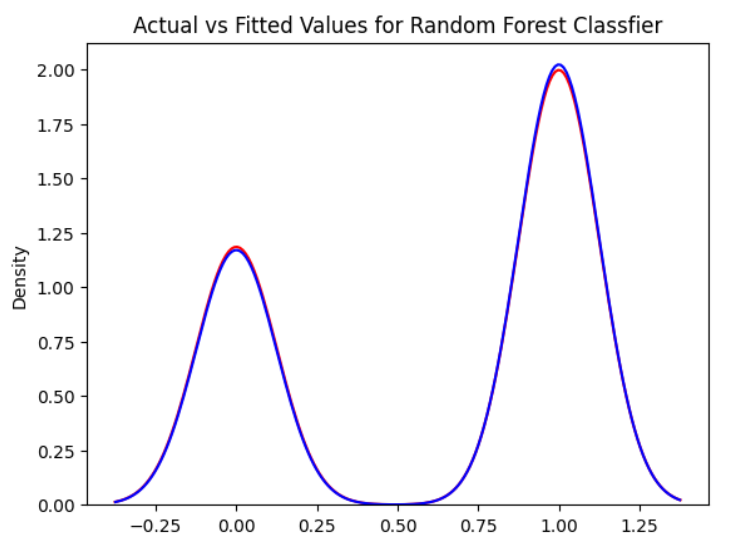


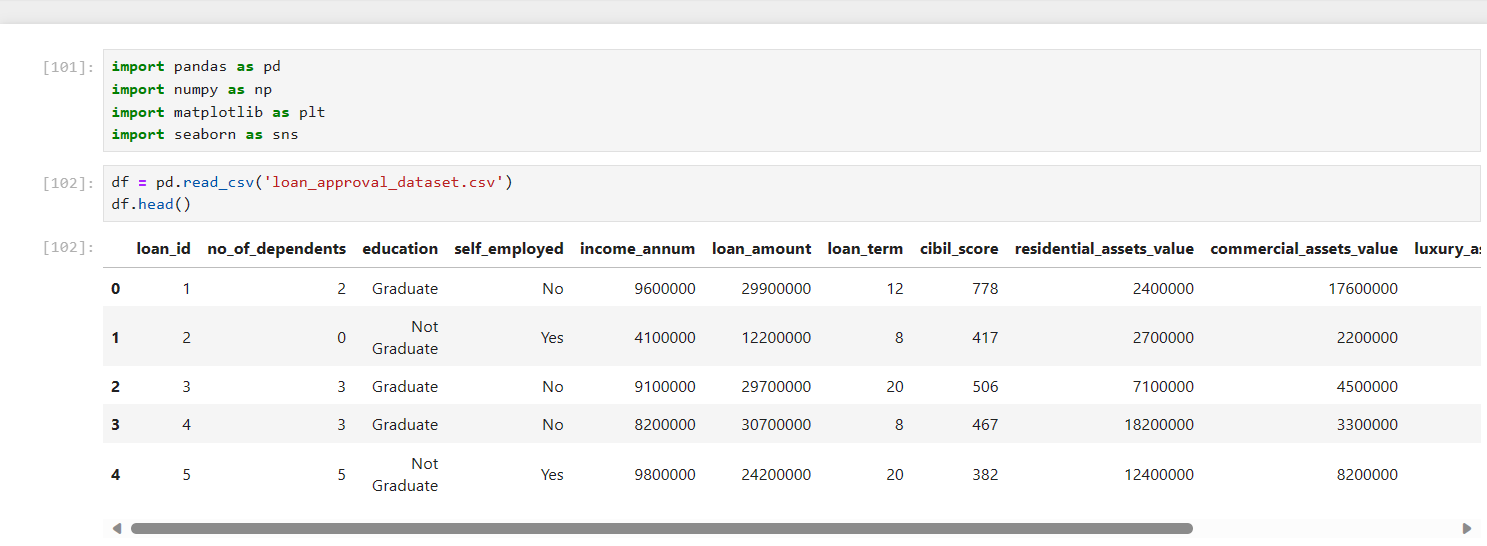


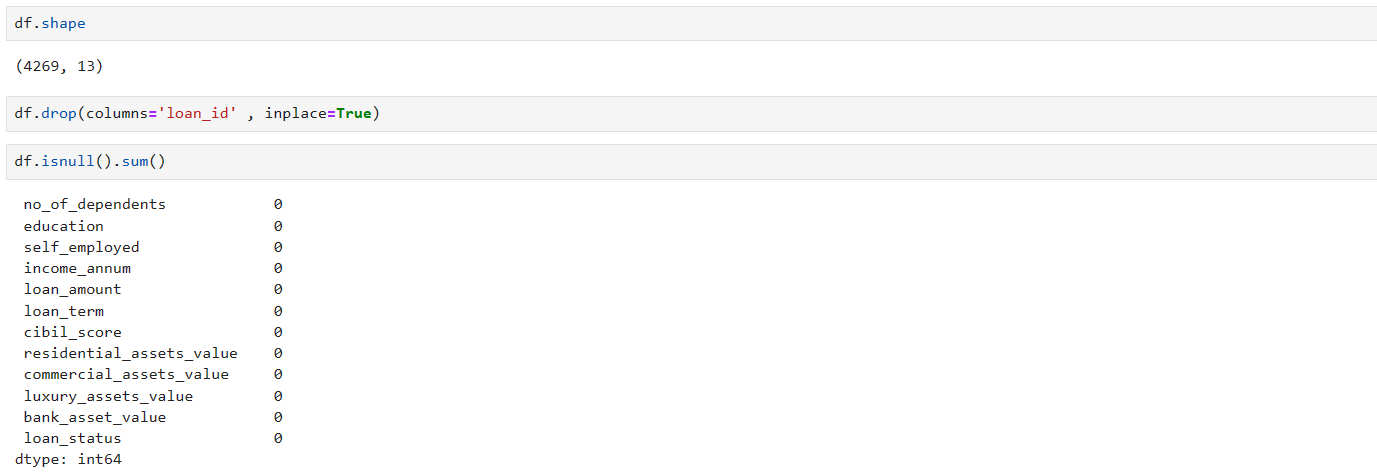
A comparison of a random and a random graph

Description automatically generated with medium confidence

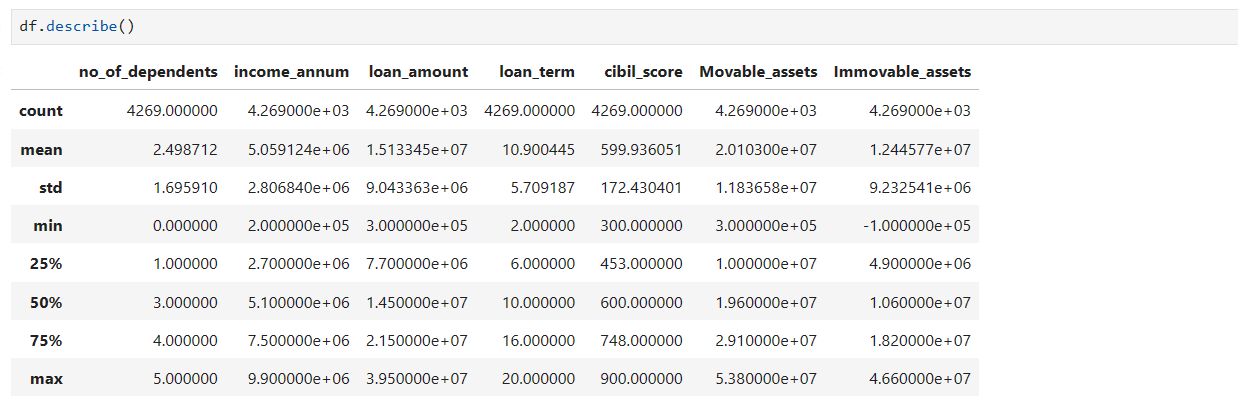


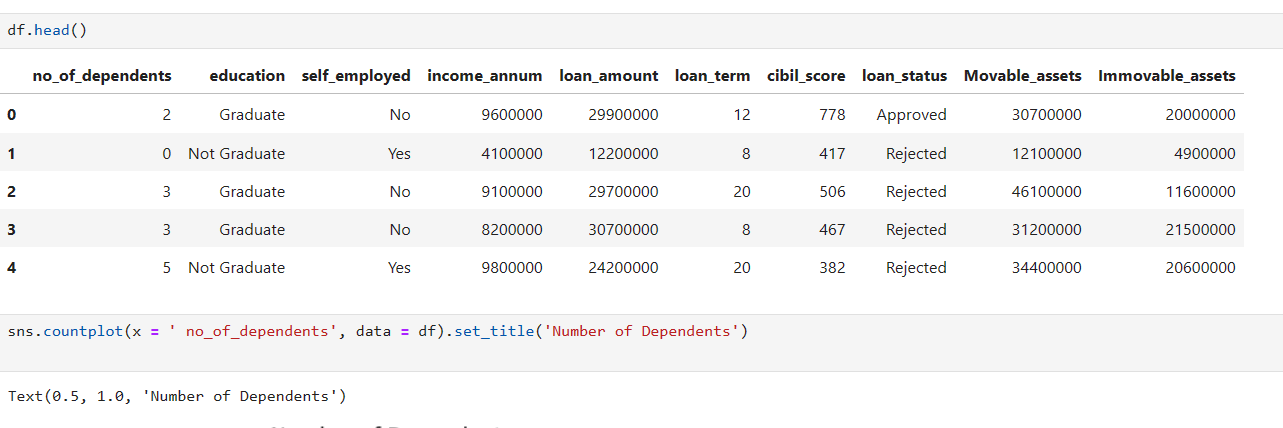


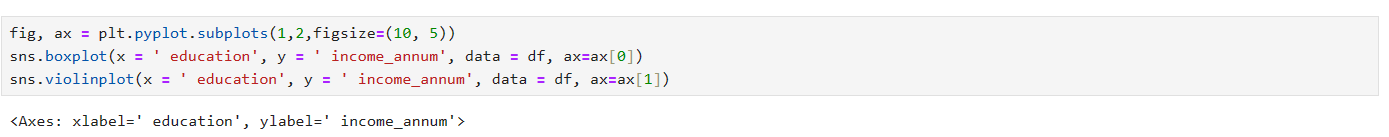


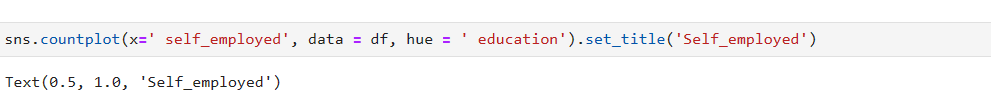


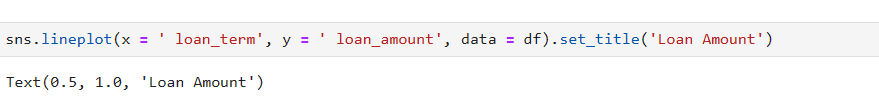


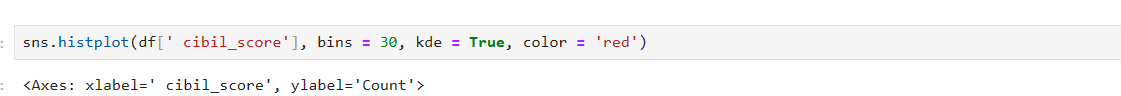


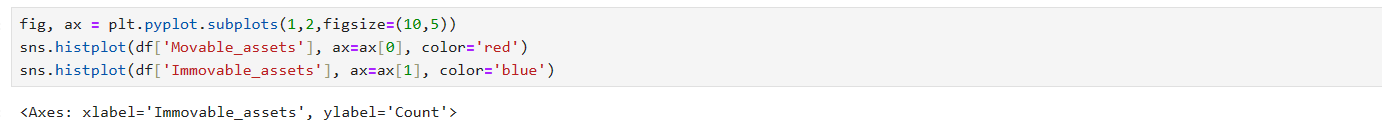




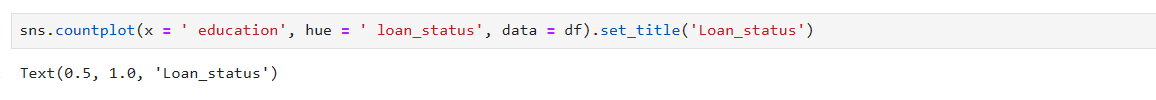


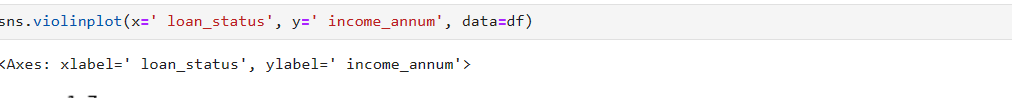


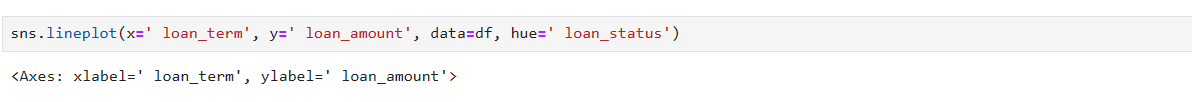








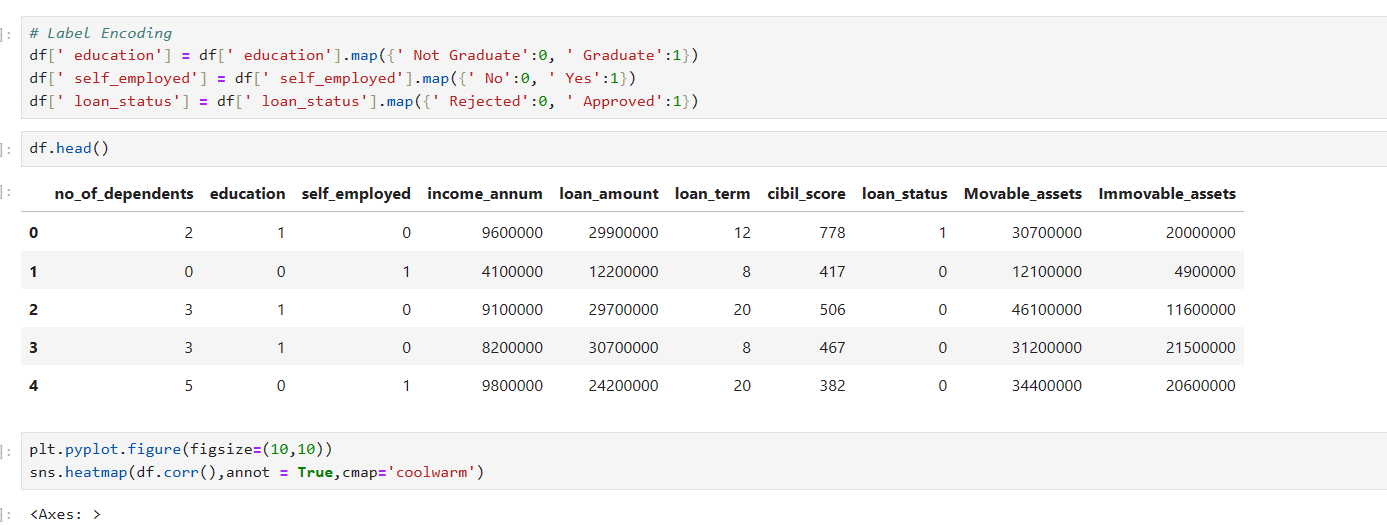


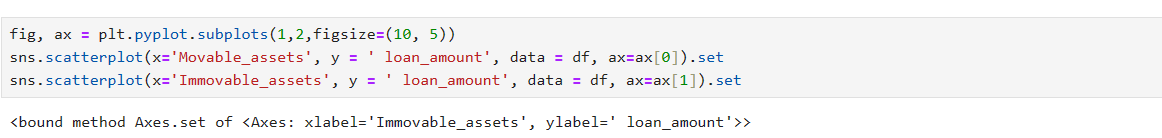


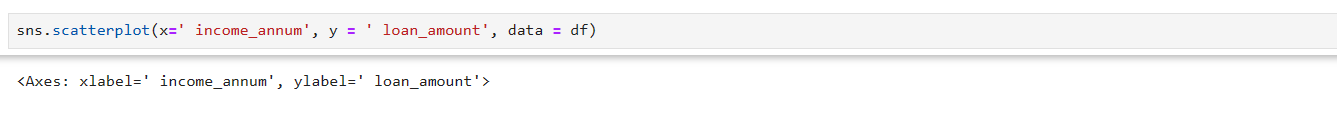


A white rectangular object with colorful text

Description automatically generated







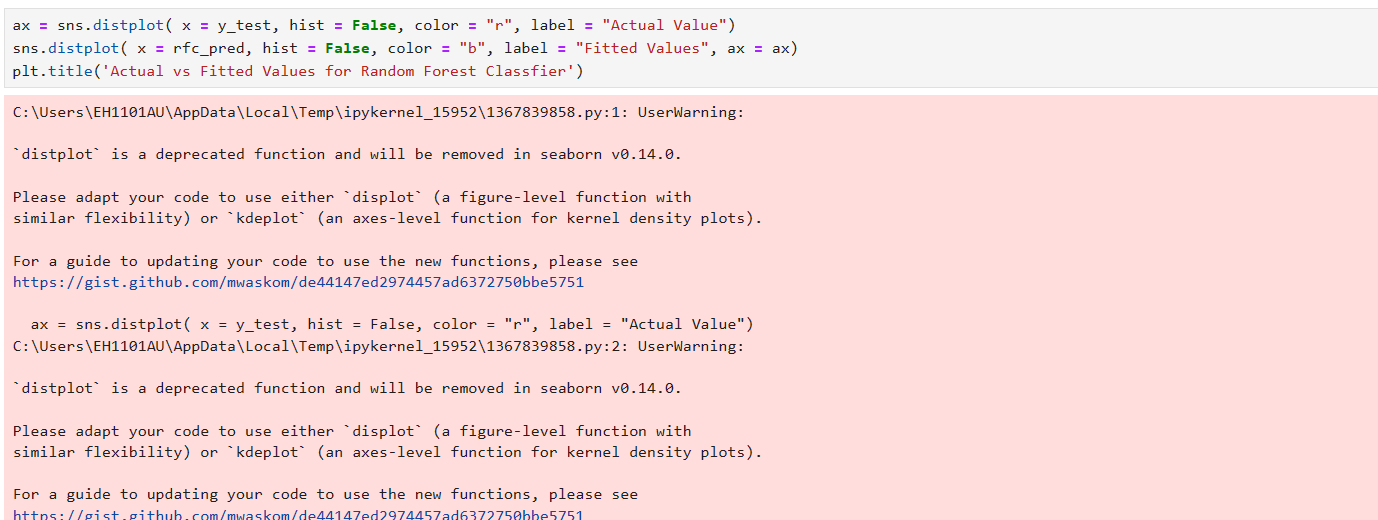


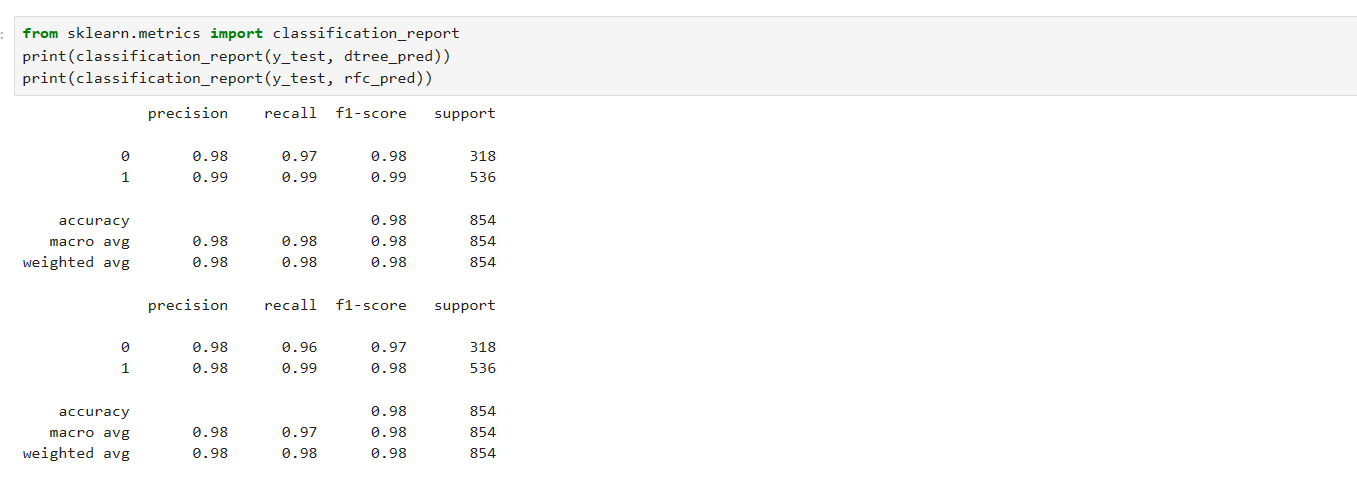




A screenshot of a computer

Description automatically generated





**11. CONCLUSION**

From the exploratory data analysis, we can conclude that the following factors are important for the approval of loan:

CIBIL Score: People with higher CIBIL score have higher chances of loan approval.

Number of Dependents: People with more number of dependents have less chances of loan approval Assets: People with more assets (including movable and immovable) have higher chances of loan approval

Loan Amount and Tenure: People with higher loan amount and lower tenure have more chances of loan approval.

Coming to the machine learning models, I have used Decision Tree Classifier and Random Forest Classifier. Both the models have given excellent results having accuracies - 91.4 % and 89.4 % respectively. But the decision tree classifier has yielded better results than the random forest classifier.

**REFERENCES**

1] Amruta S. Aphale and R. Prof. Dr. Sandeep. R Shinde, “Predict Loan Approval in Banking System Machine Learning Approach for Cooperative Banks Loan Approval”, International Journal of Engineering Trends and Applications (IJETA), vol. 9, issue 8, 2020)

[2] Loan Prediction Using Ensemble Technique, International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 3, March 2016

[3] Exploratory data analysis <https://en.wikipedia.org/wiki/Exploratory_data_analysis>

[4] Pandas Library <https://pandas.pydata.org/pandas-docs/stable/>

[5] Mean Decrease Accuracy https://dinsdalelab.sdsu.edu/metag.stats/code/randomforest.hm