GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

LOAN PREDICTION SYSTEM

A Report for the Evaluation 3 of Project 2

Submitted by

SOURAV KUMAR

(1613101750)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Dr. AMIT KUMAR GOEL, M.Tech., Ph.D.,

Professor

APRIL / MAY- 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"LOAN PREDICTION SYSTEM"** is the bonafide work of "**SOURAV KUMAR (1613101750)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Dr. AMIT KUMAR GOEL, M.Tech., Ph.D.,
Professor
School of Computing Science & Engineering

TABLE OF CONTENTS

ABSTRACT

When any financial institution lends the money to the person, it is always been a high risk. Today data is increasing with the rapid pace in the banks, therefore the bankers need to evaluate the person's data before giving the loan. It can be a big headache to evaluate the data. This problem is solved by analyzing and training the data by using one of the Machine Learning algorithms. For this, we have generated a model for the prediction that the person will get the loan or not. The primary objective of this paper is to check whether the person can get the loan or not by evaluating the data with the help of decision tree classifiers which can gives the accurate result for the prediction.

Keywords—Loan, Machine Learning, Decision tree, Data training

1. INTRODUCTION

1.1 General

The immense increase in capitalism, the fast-paced development and instantaneous changes in the lifestyle has us in awe. Emi, loans at nominal rate, housing loans, vehicle loans, these are some of the few words which have skyrocketed from the past few years. The needs, wants and demands have never been increased this before. People gets loan from banks; however, it may be baffling for the bankers to judge who can pay back the loan nevertheless the bank shouldn't be in loss. Banks earn most of their profits through the loan sanctioning. Generally, banks pass loan after completing the numerous verification processes despite all these, it is still not confirmed that the borrower will pay back the loan or not. To get over the dilemma, I have built up a prediction model which says if the loan has been assigned in the safe hands or not. Government agencies like keep under surveillance why one person got a loan and the other person could not. In Machine Learning techniques which include classification and prediction can be applied to conquer this to a brilliant extent. Machine learning has eased today's world by developing these prediction models. Here we will be using the fine techniques of machine learning – Decision tree algorithm to build this prediction model for loan assessment. It is as so because decision tree gives accuracy in the prediction and is often used in the industry for these models.

1.2 MACHINE LEARNING

1.2.1 General

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would

expect.

Fig 1. Machine Learning

The term Machine Learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence and stated that “it gives computers the ability to learn without being explicitly programmed”.

And in 1997, Tom Mitchell gave a “well-posed” mathematical and relational definition that “A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine learning involves a computer to be trained using a given data set, and use this training to predict the properties of a given new data. For example, we can train a computer by feeding it 1000 images of cats and 1000 more images which are not of a cat, and tell each time to the computer whether a picture is cat or not. Then if we show the computer a new image, then from the above training, the computer should be able to tell whether this new image is a cat or not.

Let’s try to understand Machine Learning in layman terms. Consider you are trying to toss a paper to a dustbin.

After first attempt, you realize that you have put too much force in it. After second attempt, you realize you are closer to target but you need to increase your throw angle.

What is happening here is basically after every throw we are learning something and improving the end result. We are programmed to learn from our experience.

This implies that the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing's proposal in his paper "Computing Machinery and Intelligence", in which the question "Can machines think?" is replaced with the question "Can machines do what we (as thinking entities) can do?"
Within the field of data analytics, machine learning is used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data set(input).

Suppose that you decide to check out that offer for a vacation . You browse through the travel agency website and search for a hotel. When you look at a specific hotel, just below the hotel description there is a section titled "You might also like these hotels". This is a common use case of Machine Learning called "Recommendation Engine". Again, many data points were used to train a model in order to predict what will be the best hotels to show you under that section, based on a lot of information they already know about you.

So if you want your program to predict, for example, traffic patterns at a busy intersection (task T), you can run it through a machine learning algorithm with data about past traffic patterns (experience E) and, if it has successfully "learned", it will then do better at predicting future traffic patterns (performance measure P).
The highly complex nature of many real-world problems, though, often means that inventing specialized algorithms that will solve them perfectly every time is impractical, if not impossible. Examples of machine learning problems include, "Is this cancer?", "Which of these people are good friends with each other?", "Will this person like this

movie?" such problems are excellent targets for Machine Learning, and in fact machine learning has been applied such problems with great success.

When do we need Machine Learning?

When do we need machine learning rather than directly program our computers to carry out the task at hand? Two aspects of a given problem may call for the use of programs that learn and improve on the basis of their "experience": the problem's complexity and the need for adaptivity.

Tasks That Are Too Complex to Program.

• Tasks Performed by Animals/Humans: There are numerous tasks that we human beings perform routinely, yet our introspection concerning how we do them is not sufficiently elaborate to extract a well-defined program. Examples of such tasks include driving, speech recognition, and image understanding. In all of these tasks, state of the art machine learning programs, programs that "learn from their experience," achieve quite satisfactory results, once exposed to sufficiently many training examples.

• Tasks beyond Human Capabilities: Another wide family of tasks that benefit from machine learning techniques are related to the analysis of very large and complex data sets: astronomical data, turning medical archives into medical knowledge, weather prediction, analysis of genomic data, Web search engines, and electronic commerce. With more and more available digitally recorded data, it becomes obvious that there are treasures of meaningful information buried in data archives that are way too large and too complex for humans to make sense of. Learning to detect meaningful patterns in large and complex data sets is a promising domain in which the combination of programs that learn with the almost unlimited memory capacity and ever increasing processing speed of computers opens up new horizons. Adaptivity. One limiting feature of programmed tools is their rigidity – once the program has been written down n and installed, it stays unchanged. However, many tasks change over time or from one user to another. Machine learning tools – programs whose behavior adapts to their input data – offer a solution to such issues; they are, by nature, adaptive to changes in the environment they interact with. Typical successful applications of machine learning to

such problems include programs that decode handwritten text, where a fixed program can adapt to variations between the handwriting of different users; spam detection programs, adapting automatically to changes in the nature of spam e-mails; and speech recognition programs.

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 hypothesis.

- **Feature**

  A feature is an individual measurable property of our 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.

  Note: Choosing informative, discriminating and independent features is a crucial step for effective algorithms. We generally employ a feature extractor to extract the relevant features from the raw data.

- **Target (Label)**

  A target variable or label is the value to be predicted by our model. For the fruit example discussed in the features 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).

The figure shown below clears the above concepts:

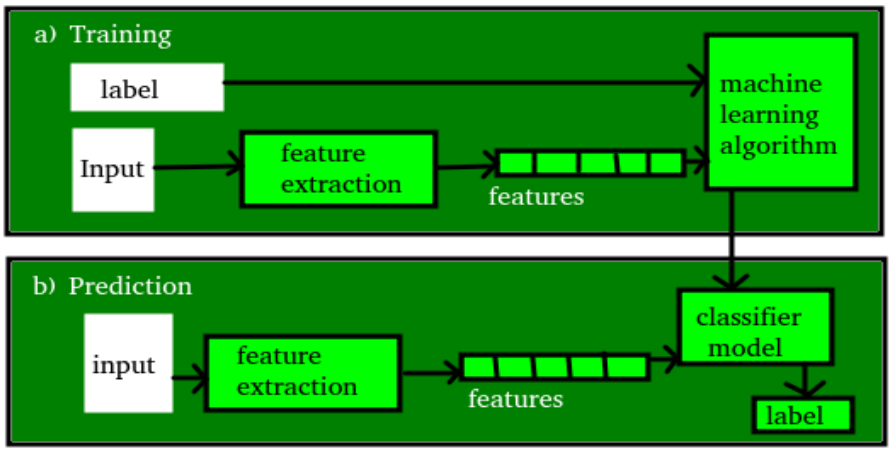


Fig-2 Training and Prediction

1.2.2 Machine Learning Types

1.2.2.1 General

Learning is, of course, a very wide domain. Consequently, the field of machine learning has branched into several subfields dealing with different types of learning tasks. We give a rough taxonomy of learning paradigms, aiming to provide some perspective of where the content sits within the wide field of machine learning.

Terms frequently used are:

- Labeled data: Data consisting of a set of training examples, where each example is a pair consisting of an input and a desired output value (also called the supervisory signal, labels, etc)
- Classification**:** The goal is to predict discrete values, e.g. {1,0}, {True, False}, {spam, not spam}.
- Regression**:** The goal is to predict continuous values, e.g. home prices.

There some variations of how to define the types of Machine Learning Algorithms but commonly they can be divided into categories according to their purpose and the main categories are the following:

- Supervised learning

- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

1.2.2.2 Supervised Learning

- I like to think of supervised learning with the concept of function approximation, where basically we train an algorithm and in the end of the process we pick the function that best describes the input
  data, the one that for a given X makes the best estimation of y (X -> y). Most of the time we are not able to figure out the true function that always make the correct predictions and other reason is that the algorithm rely upon an assumption made by humans about how the computer should learn and this assumptions introduce a bias.
- Here the human experts act as the teacher where we feed the computer with training data containing the input/predictors and we show it the correct answers (output) and from the data the computer should be able to learn the patterns.
- Supervised learning algorithms try to model relationships and dependencies between the target prediction output and the input features such that we can predict the output values for new data based on those relationships which it learned from the previous data sets.

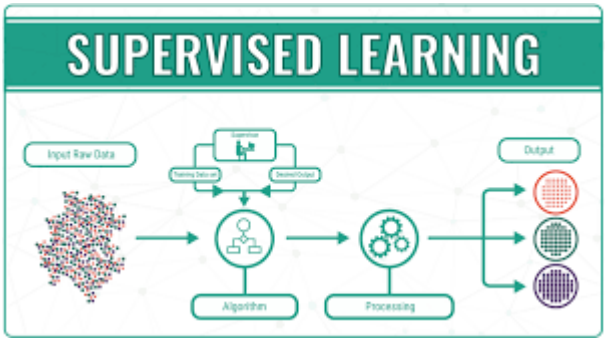


Fig 3: Supervised Learning

Draft

- Predictive Model
- we have labelled data
- The main types of supervised learning problems include regression and classification problems

List of Common Algorithms

- Nearest Neighbour
- Naive Bayes
- Decision Trees
- Linear Regression
- Support Vector Machines (SVM)
- Neural Networks

1.2.2.3 Unsupervised Learning

- The computer is trained with unlabelled data.
- Here there's no teacher at all, actually the computer might be able to teach you new things after it learns patterns in data, these algorithms a particularly useful in cases where the human expert doesn't know what to look for in the data.
- are the family of machine learning algorithms which are mainly used in pattern detection and descriptive modelling. However, there are no output categories or labels here based on which the algorithm can try to model relationships. These algorithms try to use techniques on the input data to mine for rules, detect patterns, and summarize and group the data points which help in deriving meaningful insights and describe the data better to the users.

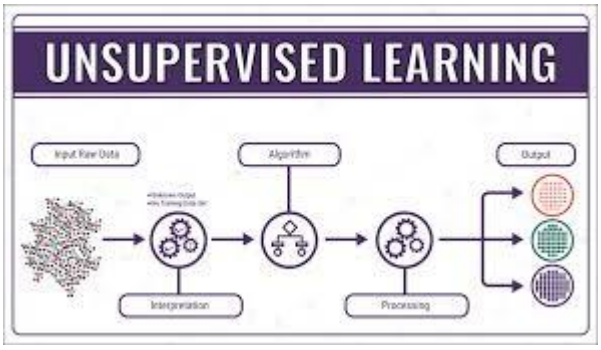


Fig 4: Unsupervised Learning

Draft

- Descriptive Model
- The main types of unsupervised learning algorithms include Clustering algorithms and Association rule learning algorithms.

List of Common Algorithms

- k-means clustering, Association Rules

1.2.2.4 Semi-Supervised Learning

In the previous two types, either there are no labels for all the observation in the dataset or labels are present for all the observations. Semi-supervised learning falls in between these two. In many practical situations, the cost to label is quite high, since it requires skilled human experts to do that. So, in the absence of labels in the majority of the observations but present in few, semi-supervised algorithms are the best candidates for the model building. These methods exploit the idea that even though the group memberships of the unlabeled data are unknown, this data carries important information about the group parameters.

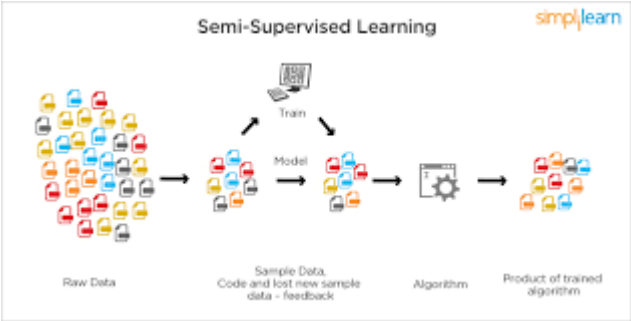


Fig 5: Semi-Supervised Learning

1.2.2.5 Reinforcement Learning

method aims at using observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk. Reinforcement learning algorithm (called the agent) continuously learns from the environment in an iterative fashion. In the process, the agent learns from its experiences of the environment until it explores the full range of possible states.

Reinforcement Learning is a type of Machine Learning, and thereby also a branch of Artificial Intelligence. It allows machines and software agents to automatically determine the ideal behaviour within a specific context, in order to maximize its performance. Simple reward feedback is required for the agent to learn its behaviour; this is known as the reinforcement signal.

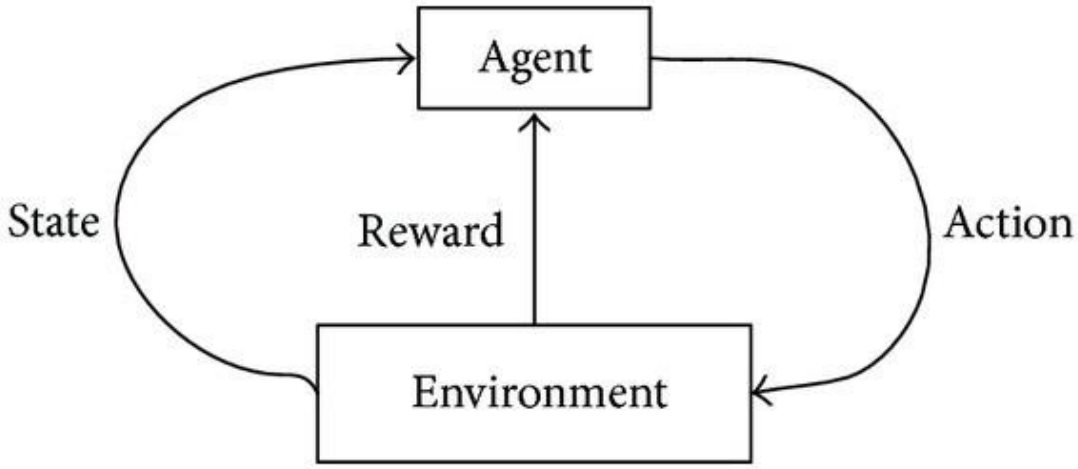


Fig 6: Reinforcement Learning

There are many different algorithms that tackle this issue. As a matter of fact, Reinforcement Learning is defined by a specific type of problem, and all its solutions are classed as Reinforcement Learning algorithms. In the problem, an agent is supposed decide the best action to select based on his current state. When this step is repeated, the problem is known as a Markov Decision Process.

In order to produce intelligent programs (also called agents), reinforcement learning goes through the following steps:

1. Input state is observed by the agent.
2. Decision making function is used to make the agent perform an action.
3. After the action is performed, the agent receives reward or reinforcement from the environment.
4. The state-action pair information about the reward is stored.

List of Common Algorithms

- Q-Learning
- Temporal Difference (TD)
- Deep Adversarial Networks

Use cases:

Some applications of the reinforcement learning algorithms are computer played board games (Chess, Go), robotic hands, and self-driving cars.

1.3 DECISION TREE

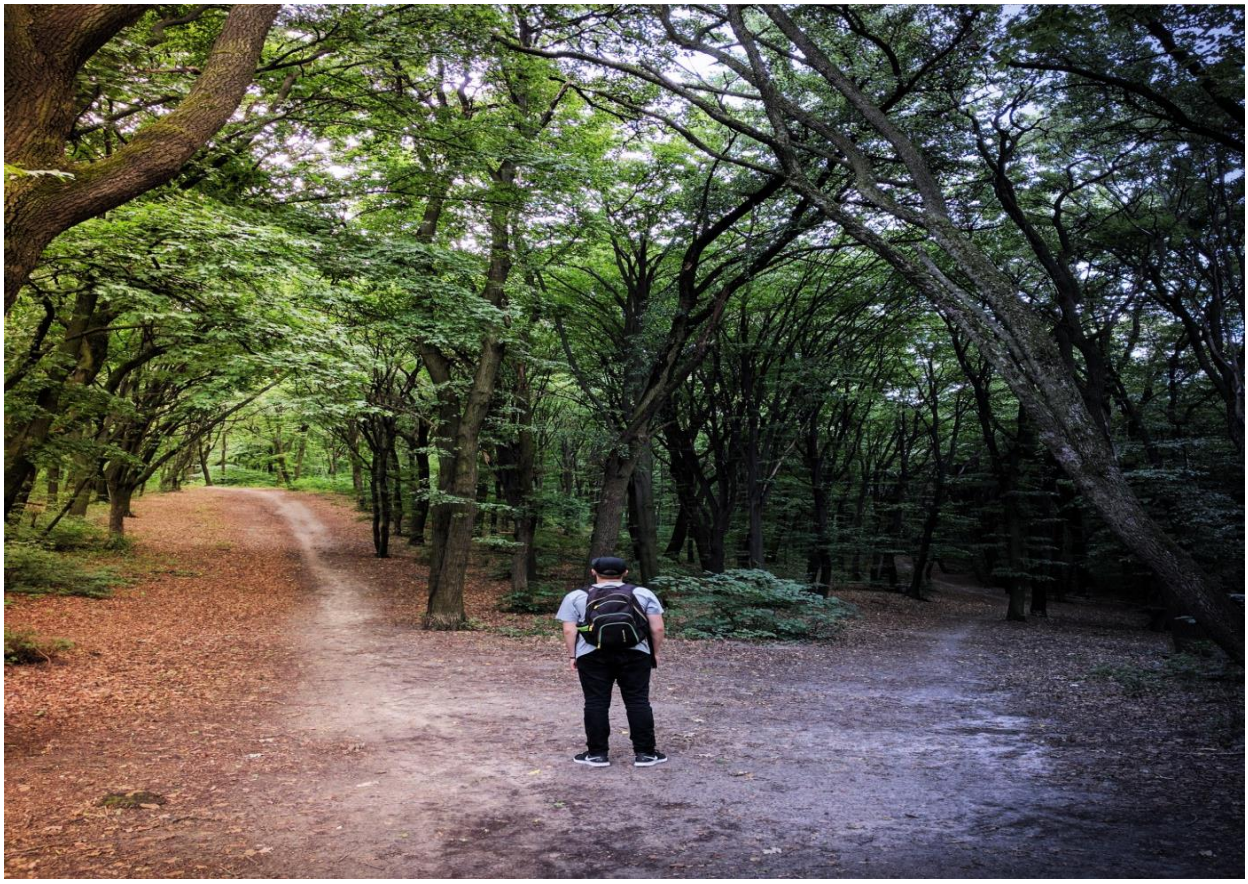


Fig 7: Decide whether to go?

A decision tree is a flowchart-like structure in which each internal node represents a test on a feature (e.g. whether a coin flip comes up heads or tails) , each leaf node represents a class label (decision taken after computing all features) and branches represent conjunctions of features that lead to those class labels. The paths from root to leaf represent classification rules. Below diagram illustrate the basic flow of decision tree for decision making with labels (Rain (Yes), No Rain (No)).

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical

methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks.

Tree models where the target variable can take a discrete set of values are called classification trees. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Classification and Regression Tree (CART) is general term for this.

Data Format

Data comes in records of forms.
(x,Y)=(x1,x2,x3,....,xk,Y)

The dependent variable, Y, is the target variable that we are trying to understand, classify or generalize. The vector x is composed of the features, x1, x2, x3 etc., that are used for that task.

Example

```
training_data = [
        ['Green', 3, 'Apple'],
        ['Yellow', 3, 'Apple'],
        ['Red', 1, 'Grape'],
        ['Red', 1, 'Grape'],
        ['Yellow', 3, 'Lemon'],
         ]
    # Header = ["Color", "diameter", "Label"]
    # The last column is the label.
    # The first two columns are features.
```

In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain
2. Gini Index

1. Information Gain

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

Entropy

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.

Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

2. Gini Index

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower Gini index should be preferred.
- Sklearn supports “Gini” criteria for Gini Index and by default, it takes “gini” value.
- The Formula for the calculation of the of the Gini Index is given below.

$$GiniIndex = 1 - \sum_j p_j^2$$

The most notable types of decision tree algorithms are: -

1. Iterative Dichotomiser 3 (ID3): This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.
2. C4.5: This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.
3. Classification and Regression Tree(CART): It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

Steps for Making decision tree

- Get list of rows (dataset) which are taken into consideration for making decision tree (recursively at each nodes).
- Calculate uncertainty of our dataset or Gini impurity or how much our data is mixed up etc.
- Generate list of all question which needs to be asked at that node.
- Partition rows into True rows and False rows based on each question asked.
- Calculate information gain based on Gini impurity and partition of data from previous step.
- Update highest information gain based on each question asked.
- Update best question based on information gain (higher information gain).
- Divide the node on best question. Repeat again from step 1 again until we get pure node (leaf nodes).

Building Decision Tree

Let’s build decision tree based on training data.

```
training_data = [
        ['Green', 3, 'Apple'],
        ['Yellow', 3, 'Apple'],
        ['Red', 1, 'Grape'],
        ['Red', 1, 'Grape'],
        ['Yellow', 3, 'Lemon'],
        ]
  # Header = ["Color", "diameter", "Label"]
  # The last column is the label.
  # The first two columns are features.

 my_tree = build_tree(training_data)

print_tree(my_tree)
```

Output

```
Is diameter >= 3?
 --> True:
Is color == Yellow?
 --> True:
Predict {'Lemon': 1, 'Apple': 1}
--> False:
Predict {'Apple': 1}
--> False:
Predict {'Grape': 2}
```

From above output we can see that at each steps data is divided into True and False rows. This process keeps repeated until we reach leaf node where information gain is 0 and further split of data is not possible as nodes are Pure.

How to avoid overfitting the Decision tree model

Overfitting is one of the major problems for every model in machine learning. If model is overfitted it will poorly be generalized to new samples. To avoid decision tree from overfitting we remove the branches that make use of features having low importance. This method is called as Pruning or post-pruning. This way we will reduce the complexity of tree, and hence improves predictive accuracy by the reduction of overfitting.

Pruning should reduce the size of a learning tree without reducing predictive accuracy as measured by cross-validation set. There are 2 major Pruning techniques.

- Minimum Error*:* The tree is pruned back to the point where the cross-validated error is a minimum.
- Smallest Tree*:* The tree is pruned back slightly further than the minimum error. Technically the pruning creates a decision tree with cross-validation error within 1 standard error of the minimum error.

Early Stop or Pre-pruning

An alternative method to prevent overfitting is to try and stop the tree-building process early, before it produces leaves with very small samples. This heuristic is known as early stopping but is also sometimes known as pre-pruning decision trees.

At each stage of splitting the tree, we check the cross-validation error. If the error does not decrease significantly enough then we stop. Early stopping may underfit by stopping too early. The current split may be of little benefit, but having made it, subsequent splits more significantly reduce the error.

Early stopping and pruning can be used together, separately, or not at all. Post pruning decision trees is more mathematically rigorous, finding a tree at least as good as early

stopping. Early stopping is a quick fix heuristic. If used together with pruning, early stopping may save time. After all, why build a tree only to prune it back again?

Important Terminology related to Decision Trees

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

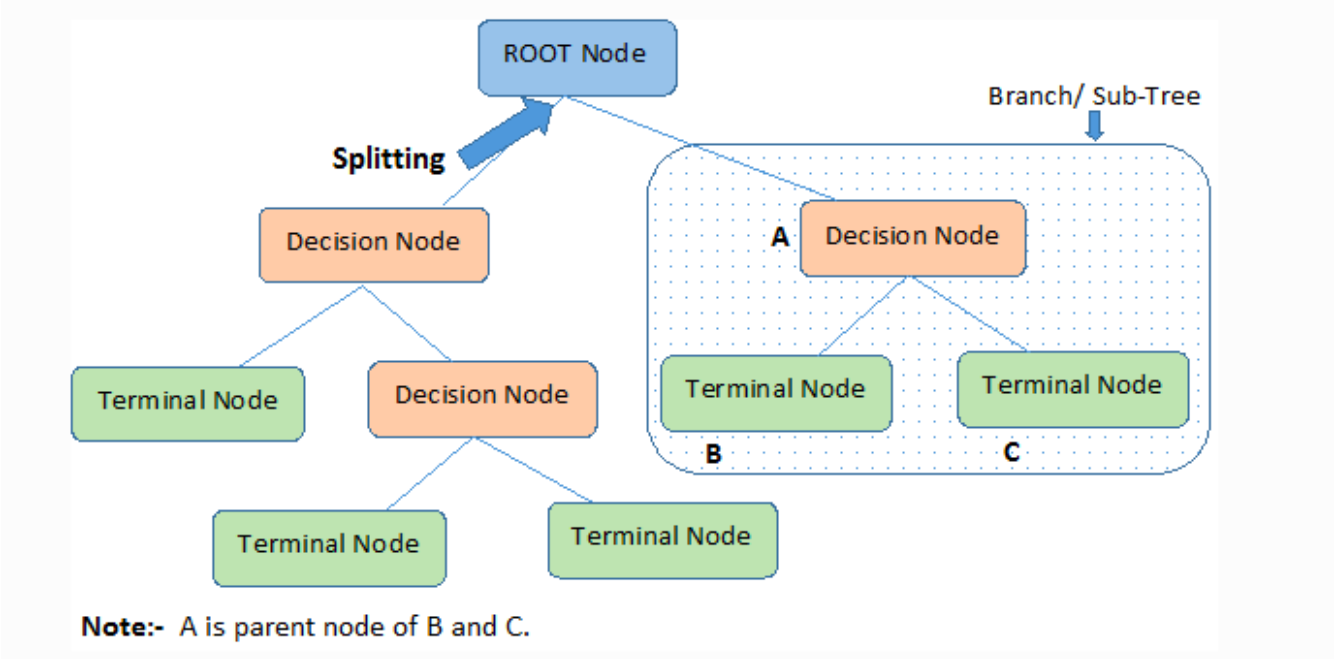


Fig 8: Decision Tree Terminlogy Representation

Decision trees classify the examples by sorting them down the tree from the root to some leaf/terminal node, with the leaf/terminal node providing the classification of the example.

Each node in the tree acts as a test case for some attribute, and each edge descending from the node corresponds to the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new node.

Optimizing the Decision Tree Classifier

- **criterion**: optional (default=”gini”) or Choose attribute selection measure: This parameter allows us to use the different-different attribute selection measure. Supported criteria are “gini” for the Gini index and “entropy” for the information gain.
- **splitter**: string, optional (default=”best”) or Split Strategy: This parameter allows us to choose the split strategy. Supported strategies are “best” to choose the best split and “random” to choose the best random split.
- **max_depth**: int or None, optional (default=None) or Maximum Depth of a Tree: The maximum depth of the tree. If None, then nodes are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting (Source).

In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. The maximum depth of the tree can be used as a control variable for pre-pruning.

```
# Create Decision Tree classifer object
classifier = DecisionTreeClassifier (criterion="entropy", max_depth=3)
# Train Decision Tree Classifer
classifier = classifier.fit(X_train,y_train)
#Predict the response for test dataset
y_pred = classifier.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Advantage of Decision Tree

- Easy to use and understand.

- Can handle both categorical and numerical data.
- Resistant to outliers, hence require little data pre-processing.

Disadvantage of Decision Tree

- Prone to overfitting.
- Require some kind of measurement as to how well they are doing.
- Need to be careful with parameter tuning.
- Can create biased learned trees if some classes dominate.

2. LITERATURE SURVEY

2.1 General

Machine learning helps to learn data from its own experience and in prediction and decision of data. It has brought the revolution in the field of computer science because data is the most important thing in the world. To analyze data, Machine Learning has given many solutions by its algorithms. I have gone through many papers before developing this model to collect information. In paper (1), authors have tried to reduce the efforts of banks by generating a model by various machine learning models and explained which of the methods can be accurate. The paper was divided into 4 sections- data collection, comparison of various machine learning models on data, training of data and testing. They have done by the mining the previous data. Author has also mentioned that this system can be integrated with the Automatic-processing system in future. In paper (2), authors basically want to know about the nature of client by analyzing various variables. In this paper, author compares various attributes by exploratory data analysis technique, seven graphs were generated and short-term loan was preferred in the paper. In paper (3), the authors have explained about whether it is safe or not to give loans to the person. They have used map function to predict the records. In paper (4), authors have used decision tree induction algorithm to implement a model and tried to review credit scoring of mortgage loans and criteria for the applicants. Credit score helps in sanctioning of the loan. They have developed a model to predict it is safe or not for loan sanctioning and it was concluded that mostly low-income applicants receive loan approval as they are likely to repay their loan back. Dataset is collected from the Kaggle. In paper (6), authors have tried to evaluate the credit risks and to identify the loan repayment prediction. They have made use of decision tree algorithm in the paper. A test set is utilized to approve the form. In paper (7), authors have used data mining to develop a model and the system has 3 components- preprocessing, classification and database updation. KNN, Binning and naïve-Bayesian algorithm was used for prediction model. A hybrid of naïve-Bayesian and K-means was used for improving the

efficiency of the system. In paper (8), authors have used three types of machine algorithm which are Logistic Regression, Decision tree and Random Forest algorithm. After analyzing the data sets on these models, Random Forest algorithm come out with the highest accuracy of all the model. The main aim of this model was to sanction loan to applicants in a short span of time. In paper (9), an ensemble model is used by the authors which gives better accuracy and efficiency than the individual models for prediction. In paper (10), authors have used classifier technique in which combination of min-max normalization and KNN algorithm is used which gives 75.08% accuracy result. R programming language is used to build KNN soring credit model. This paper focuses on predicting the loan status in commercial bank.

It cannot be clear which model is best because each model has its own specification. Therefore, an efficient model is important to be developed which can give higher accuracy.

3. IMPLEMENTATION OF MODEL

3.1 EXISTING SYSTEM

Banks need to analyze for the person who applies for the loan will repay the loan or not. Sometime it happens that customer has provided partial data to the bank, in this case person may get the loan without proper verification and bank may end up with loss. Bankers cannot analyze the huge amounts of data manually, it may become a big headache to check whether a person will repay its loan or not. It is very much necessary to know the person getting loan is going in safe hand or not. So, it is pretty much important to have a automated model which should predict the customer getting the loan will repay the loan or not.

3.2 PROPOSED SYSTEM

I have developed a prediction model for Loan sanctioning which will predict whether the person applying for loan will get loan or not. The major objective of this project is to derive patterns from the datasets which are used for the loan sanctioning process and create a model based on the patterns derived in the previous step. This model is developed by using the one of the machine learning algorithms.

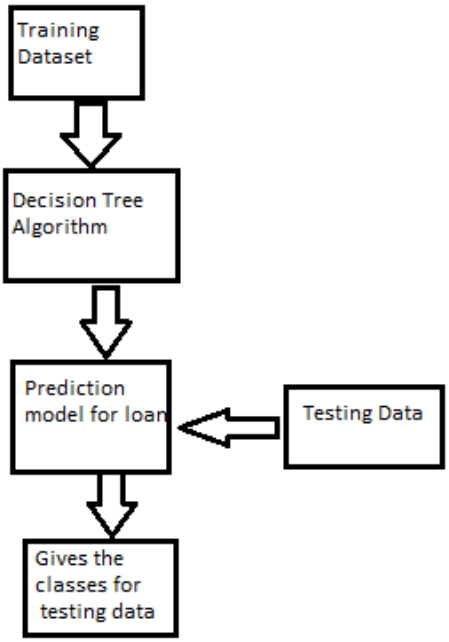


Fig 9: Architecture of propsed system

In the proposed model for loan prediction, Dataset is split into training and testing data. After then training datasets are trained using the decision tree algorithm and a prediction model is developed using the algorithm. Testing datasets are then given to model for the prediction of loan. The motive of this paper is to predict the defaults who will repay the loan or not. Various libraries like pandas, numpy have been used. After the loading of datasets, Data Preprocessing like missing value treatment of numerical and categorical is done by checking the values. Numerical and categorical values are segregated. Outliers and frequency analysis are done, outliers are checked by getting the boxplot diagram of attributes.

3.2.1 DATA SET:

Datasets are gathered from Kaggle. Data set is now provided to Machine learning models on the basis of this facts this version is trained. Data sets are divided into Existing and New Customers. Every new applicant info act as a fact test set. After the operation of testing, model expect whether the brand-new applicant is in case for approval of the loan or now not primarily based upon the inference it concludes on the idea of training information sets.

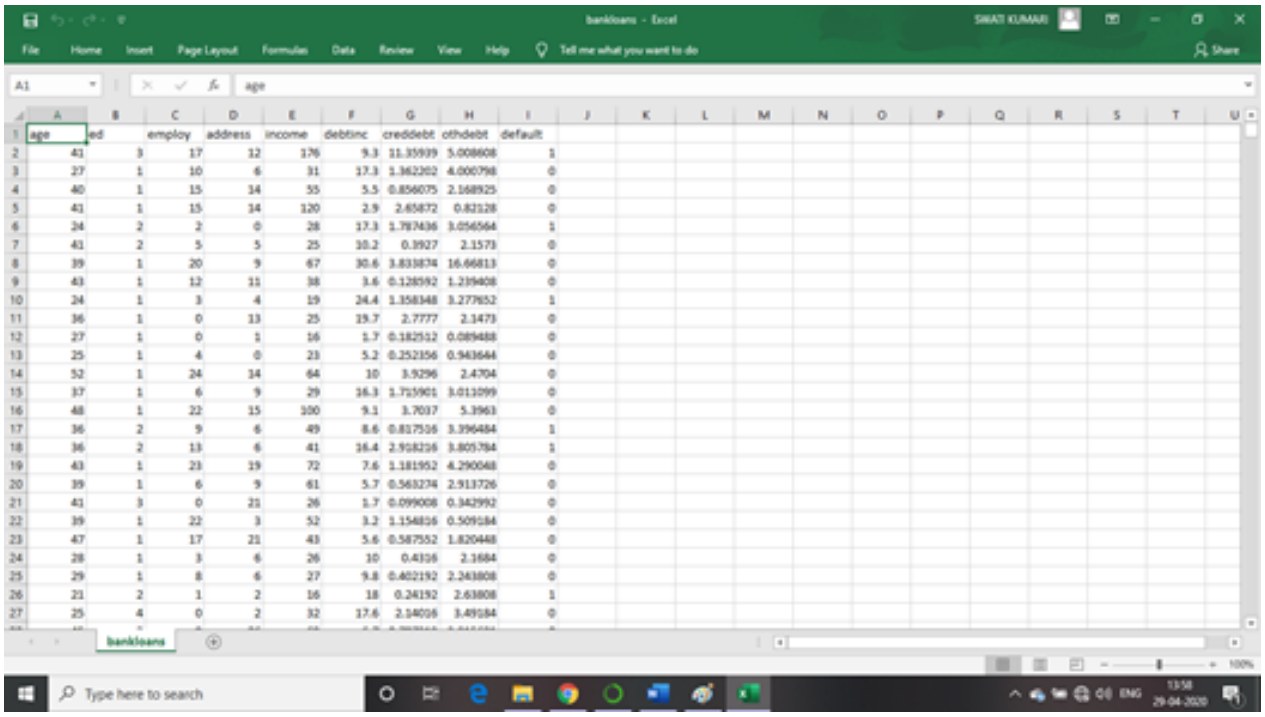


Fig 10: Datasets from Kaggle

3.3 IMPLEMENTATION OF MODEL

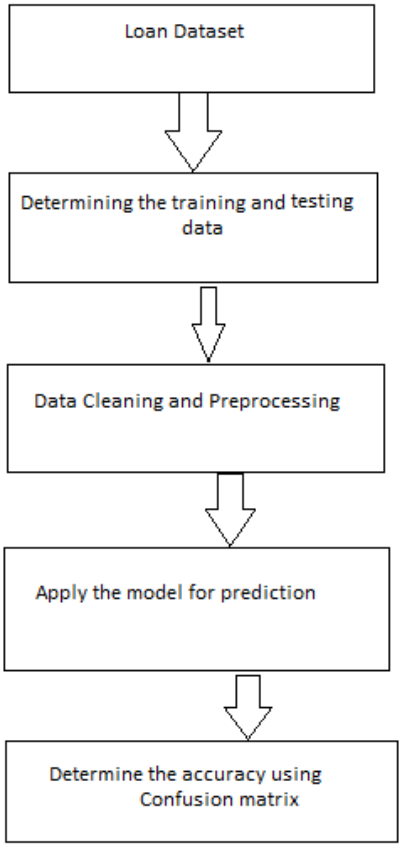


Fig 11: Diagram of Prediction Model

3.3.1 Data Exploratory Analysis

Data Exploratory Analysis is done through Bivariate Analysis by Numeric (TTest) or Categorical (Chisquare). Visualization of Attributes are also done by Bivariate Analysis. We use a BivariateAnalysisPlot which is used to analyze the impact of features on the target variables.

Following are the bivariable analysis of attributes:

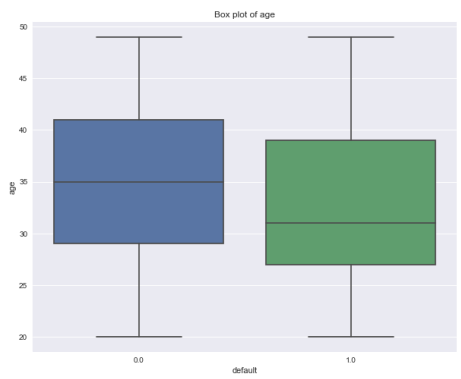


Fig 12: Box plot of age

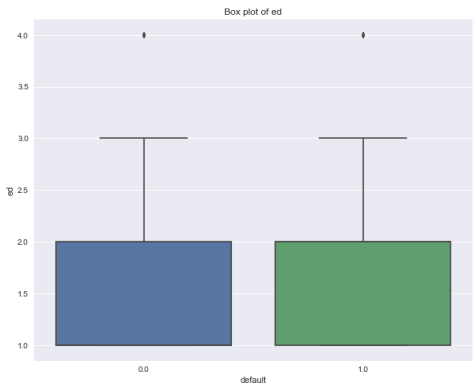


Fig 13: Boxplot of education

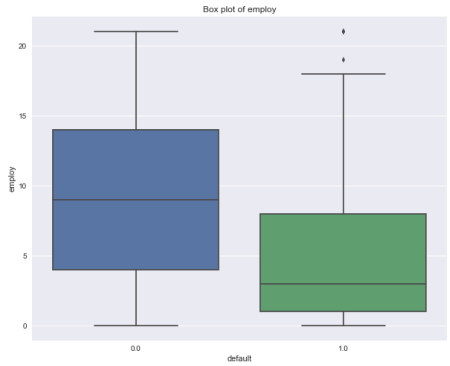


Fig 14: Boxplot of employ

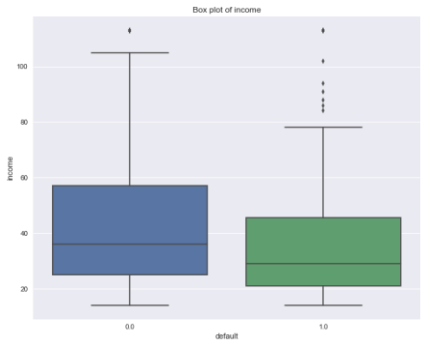


Fig 15: Boxplot of income

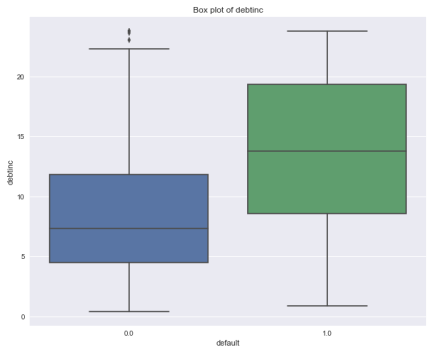


Fig 16: Boxplot of debtincome

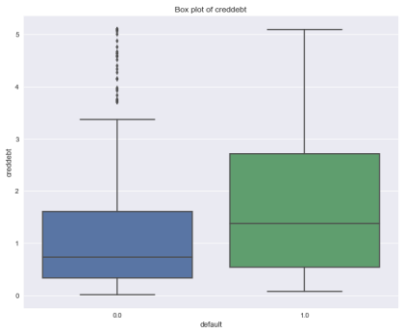


Fig 17: Boxplot of creditdebt

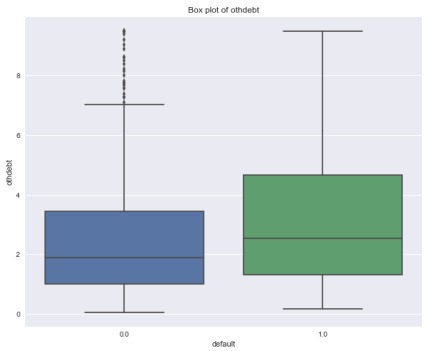


Fig 18: Boxplot of otherdebt

In above exploratory analysis of features, it basically shows the dependency or association of features of defaults and non- default attributes. In fig 2, graph shows the relationship of age between defaulters and non-defaulters and shows the age group of them. Similarly, other plots of features also show connection between them. Suppose, if someone is already in debt and income of customer is not that much, the probability of repaying his loan will be low.

There are 850 observations and 9 features in the data set - All 9 features are numerical in

nature - There are no missing values in the data set - Out of 850 customers data, 700 are existing customers and 150 are new customers - In the 700 existing customers, 517 customers are tagged as non-defaulters and remaining 183 are tagged as defaulters - The data is highly imbalanced - From VIF check, found out that the correlation between the variables is within the acceptable limits.

3.3.2 Decision Tree:

Decision Tree is a greedy algorithm it searches the entire space of possible decision trees. so, we need to find an optimum parameter(s) or criteria for stopping the decision tree at some point. We use the hyperparameters to prune the decision tree.

Decision tree algorithm used in model:

We have built a classification model which learn decision rules gathered from data features will make predictions and will generate a tree structure with the help of decision nodes corresponding to input variables. Following processes are followed to classify the data:

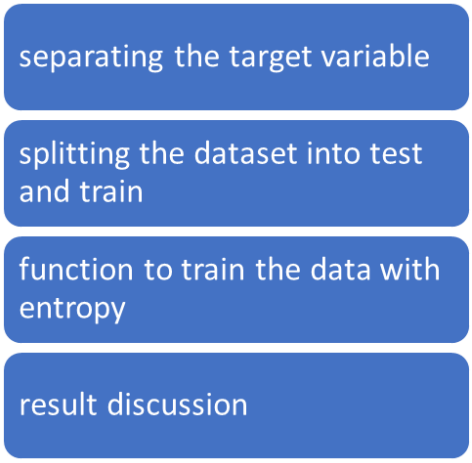
1)For selection of attribute we will be using splitting criterion Gini index. The best score of the attribute will be chosen as deciding node.

2)Root split node is created with other subsets and then first process is repeated until the next best attribute is selected as deciding node.

3)Process 2 is continued until reaching a leaf node.

4)In the last step, pruning is applied to avoid the overfitting by removing that sections of tree which has little classification power and determine the optimum size of tree.

3.3.3 Processes for Loan Prediction:



Separating the target variable

X= balance_data. values[:, 1:5]

Y= balance_data. values[:, 0]

Splitting the dataset into test and train

X_train, X_test, Y_train, y_test= train_test_split (X, Y, test_size = 0.3, random_state= 100)

Function to perform training with entropy

Clf_entropy=DecisionTreeClassifier (random_state=100, max_depth =5, min_samples_leaf =7)

Clf_entropy.fit (X_train, Y_train)

Gini Index is the splitting criterion for the tree and decides address as a root node.

3.3.4 Generated Decision Tree for Model

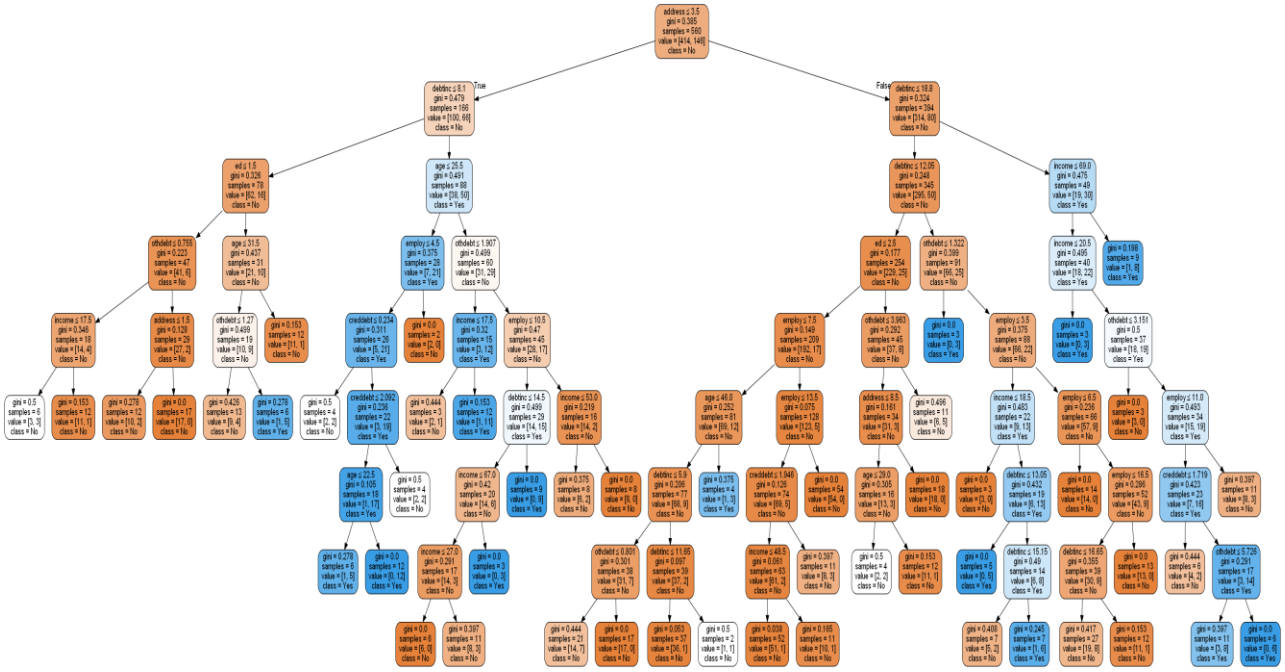


Fig 19: Decision tree for the model

Visualization of decision tree is with the help of graphviz package.

3.3.5 Source Code

Import Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
%matplotlib inline
plt.rcParams['figure.figsize'] = 10, 8
```

```
plt.style.use("seaborn")
#for machine learning
import statsmodels.formula.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_validate
```

Load DataSets

```
bankloans = pd.read_csv("Data/bankloans.csv")
bankloans.head()
bankloans.columns
#number of observations and features
bankloans.shape
#data types in the dataframe
bankloans.info()
#check for any column has missing values
bankloans.isnull().any()
#check for number of missing values
bankloans.isnull().sum()
#Segregating the numeric and categorical variable names
numeric_var_names = [key for key in dict(bankloans.dtypes) if dict(bankloans.dtypes)[key] in ['float64', 'int64', 'float32', 'int32']]
catgorical_var_names = [key for key in dict(bankloans.dtypes) if dict(bankloans.dtypes)[key] in ['object']]
numeric_var_names
#splitting the data set into two sets - existing customers and new customers
bankloans_existing = bankloans.loc[bankloans.default.isnull() == 0]
```

```
bankloans_new = bankloans.loc[bankloans.default.isnull() == 1]
bankloans_existing.describe(percentiles=[.25,0.5,0.75,0.90,0.95])
sns.boxplot(y = "age",data=bankloans_existing)
plt.title("Box-Plot of age")
plt.show()
sns.boxplot(y = "employ",data=bankloans_existing)
plt.title("Box-Plot of employee tenure")
plt.show()
sns.boxplot(y = "income",data=bankloans_existing)
plt.title("Box-Plot of employee income")
plt.show()
sns.boxplot(y = "debtinc",data=bankloans_existing)
plt.title("Box-Plot of employee debt to income ratio")
plt.show()
sns.boxplot(y = "creddebt",data=bankloans_existing)
plt.title("Box-Plot of Credit to debit ratio")
plt.show()
income_minlimit = bankloans_existing["income"].quantile(0.75) + 1.5 *
(bankloans_existing["income"].quantile(0.75) -
bankloans_existing["income"].quantile(0.25))
income_minlimit
def outlier_capping(x):
    """A funtion to remove and replace the outliers for numerical columns"""
    x = x.clip_upper(x.quantile(0.95))
    return(x)
#outlier treatment
bankloans_existing = bankloans_existing.apply(lambda x: outlier_capping(x))
##Correlation Matrix
bankloans_existing.corr()
#Visualize the correlation using seaborn heatmap
```

```
sns.heatmap(bankloans_existing.corr(),annot=True,fmt="0.2f",cmap="coolwarm")
plt.show()
bankloans_existing.shape
bankloans_new.shape
#Indicator variable unique types
bankloans_existing['default'].value_counts()
bankloans_existing['default'].value_counts().plot.bar()
plt.xlabel("default")
plt.ylabel("count")
plt.title("Distribution of default")
plt.show()
#percentage of unique types in indicator variable
round(bankloans_existing['default'].value_counts()/bankloans_existing.shape[0] * 100,3)
## performing the independent t test on numerical variables
tstats_df = pd.DataFrame()
for eachvariable in numeric_var_names:
    tstats = stats.ttest_ind(bankloans_existing.loc[bankloans_existing["default"] == 1,eachvariable],bankloans_existing.loc[bankloans_existing["default"] == 0, eachvariable],equal_var=False)
    temp = pd.DataFrame([eachvariable, tstats[0], tstats[1]]).T
    temp.columns = ['Variable Name', 'T-Statistic', 'P-Value']
    tstats_df = pd.concat([tstats_df, temp], axis=0, ignore_index=True)
tstats_df =  tstats_df.sort_values(by = "P-Value").reset_index(drop = True)
tstats_df
def BivariateAnalysisPlot(segment_by):
    """A funtion to analyze the impact of features on the target variable"""
    fig, ax = plt.subplots(ncols=1,figsize = (10,8))
    #boxplot
    sns.boxplot(x = 'default', y = segment_by, data=bankloans_existing)
    plt.title("Box plot of "+segment_by)
```

```
    plt.show()
BivariateAnalysisPlot("age")
BivariateAnalysisPlot("ed")
BivariateAnalysisPlot("employ")
BivariateAnalysisPlot("address")
BivariateAnalysisPlot("income")
BivariateAnalysisPlot("debtinc")
BivariateAnalysisPlot("creddebt")
BivariateAnalysisPlot("othdebt")
#Decision tree Classifier
#make a pipeline for decision tree model
pipelines = {
    "dtclass": make_pipeline(DecisionTreeClassifier(random_state=100))
}
#To check the accuracy of the pipeline
scores = cross_validate(pipelines['dtclass'],train_X,train_y,return_train_score=True)
scores['test_score'].mean()
#list of tunable hyper parameters for decision tree classifier pipeline
pipelines['dtclass'].get_params().keys()
decisiontree_hyperparameters = {
    'decisiontreeclassifier__max_depth' : np.arange(3, 10),
    'decisiontreeclassifier__max_features' : np.arange(3, 8),
    'decisiontreeclassifier__min_samples_split' : np.arange(2, 15),
    "decisiontreeclassifier__min_samples_leaf" : np.arange(1,3)
}
#Create a cross validation object from decision tree classifier and it's hyperparameters
dtclass_model  =  GridSearchCV(pipelines['dtclass'],decisiontree_hyperparameters,cv=5,
n_jobs=-1)
#fit the model
dtclass_model.fit(train_X, train_y)
```

```
#display the best parameters for decision tree model
dtclass_model.best_params_
#best score for the model
dtclass_model.best_score_
#In Pipeline we can use the string names to get the decisiontreeclassifer
dtclass_best_model =
dtclass_model.best_estimator_.named_steps['decisiontreeclassifier']
dtclass_best_model
#Predicting the test cases
bankloans_test_pred_dtclass = pd.DataFrame({'actual':test_y, 'predicted':
dtclass_best_model.predict(test_X)})
bankloans_test_pred_dtclass = bankloans_test_pred_dtclass.reset_index()
bankloans_test_pred_dtclass.head()
#creating a confusion matrix
cm_dtclass = metrics.confusion_matrix(bankloans_test_pred_dtclass.actual,
                    bankloans_test_pred_dtclass.predicted,labels = [1,0])
cm_dtclass
sns.heatmap(cm_dtclass,annot=True, fmt=".2f",
cmap="Greens",linewidths=.5,linecolor="red",
        xticklabels = ["Default", "Not Default"] , yticklabels = ["Default", "Not
Default"])
plt.title("Confusion Matrix for Test data")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")
plt.show()
#probabilty of prediction
predict_prob_df = pd.DataFrame(dtclass_best_model.predict_proba(test_X))
predict_prob_df.head()
bankloans_test_pred_dtclass = pd.concat([bankloans_test_pred_dtclass,
predict_prob_df], axis = 1)
```

```
bankloans_test_pred_dtclass.columns = ['index', 'actual', 'predicted', 'default_0','default_1']
bankloans_test_pred_dtclass.head()
#find the auc score
auc_score = metrics.roc_auc_score(bankloans_test_pred_dtclass.actual, bankloans_test_pred_dtclass.default_1)
round(auc_score,4)
#plotting the roc curve

fpr, tpr, thresholds = metrics.roc_curve(bankloans_test_pred_dtclass.actual, bankloans_test_pred_dtclass.default_1,
                                         drop_intermediate=False)
plt.plot(fpr, tpr, label = "ROC Curve (Area = %0.4f)" % auc_score)
plt.plot([1,0],[1,0],'k--')
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate or [1 - True Negative Rate]")
plt.legend(loc = "lower right")
plt.show()
#find precision score
prec_score = metrics.precision_score(bankloans_test_pred_dtclass.actual, bankloans_test_pred_dtclass.predicted)
print("Precision score :", round(prec_score,3))
#find the overall accuracy of model
acc_score = metrics.accuracy_score(bankloans_test_pred_dtclass.actual,bankloans_test_pred_dtclass.predicted)
print("Accuracy of model :", round(acc_score,3))
#Visualization of Decision tree
from sklearn.externals.six import StringIO
from IPython.display import Image
```

```
from sklearn.tree import export_graphviz
import pydotplus as pdot
#writing the dot data
dot_data = StringIO()
#export the decision tree along with the feature names into a dot file format
export_graphviz(dtclass_best_model,out_file=dot_data,filled=True,special_characters=T
rue,rounded=True,
                feature_names=train_X.columns.values,class_names = ["No","Yes"])
#make a graph from dot file
graph = pdot.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

4. RESULTS

The Confusion Matrix (CM) is used to analyze and determine the performance of the proposed loan prediction model.

- True Positive (TP), when both the actual and predicted values are positive (1)

- True Negative (TN), when both the actual and predicted values are negative (0)

- False Positive (FP), when the actual value is negative and the predicted value is positive (1)

- False Negative (FN), when the actual value is positive (1) and the predicted value is negative(0)

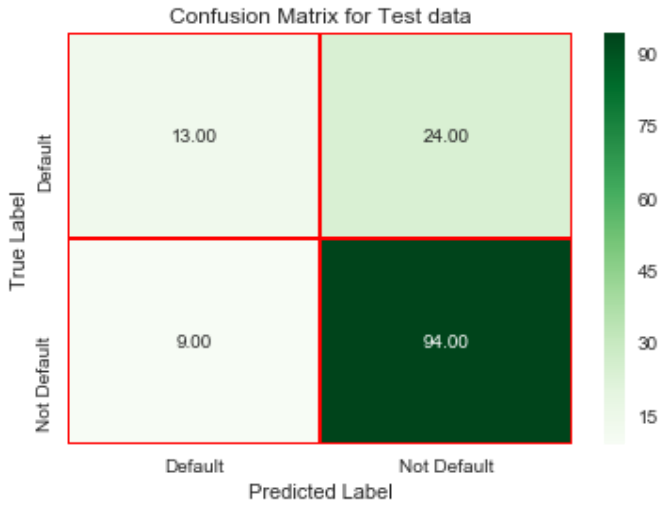


Fig 20: Generated Confusion Matrix for the Model

Confusion Matrix is given for the test data after the prediction of the model for loan. It shows the performance of the model developed for the prediction.

Accuracy for the model:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

After the model is built, best score for the model is 0.7893. Then the model performance evaluations of test datasets are done and datasets passed through the model. Precision score of the test data is 0.591. Overall Accuracy of the model is 0.764.

ROC Curve for the model

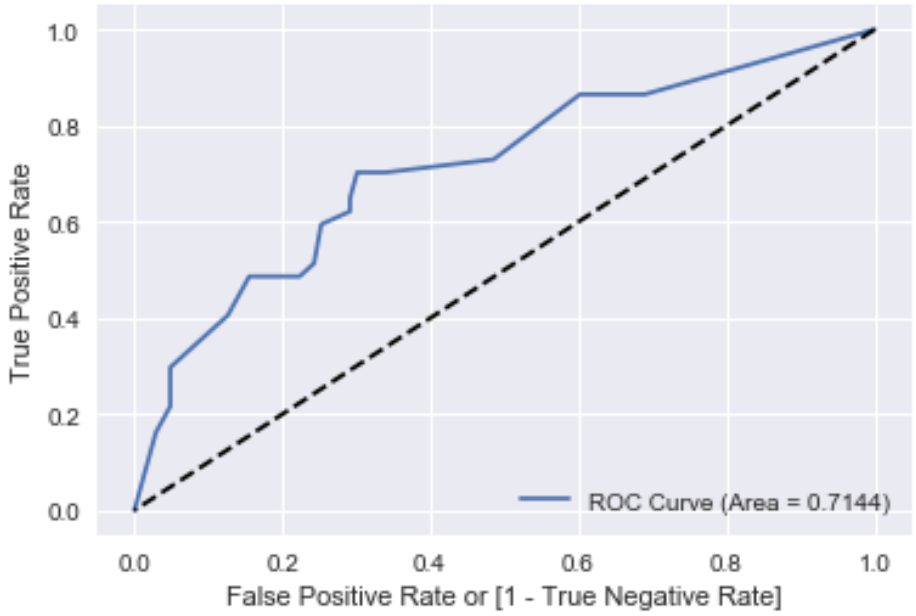


Fig 21: ROC Curve for the model

ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

5. Conclusion

5.1 Future Work

In future, this model can be used to compare various machine learning algorithm generated prediction models and the model which will give higher accuracy will be chosen as the prediction model.

5.2 Conclusion

After this work, we are able to conclude that Decision tree version is extraordinary efficient and gives a higher end result. We have developed a model which can easily predict that the person will repay its loan or not. we can see our model has reduced the efforts of bankers. Machine learning has helped a lot in developing this model which gives precise results.

6. REFERENCES

[1] Kumar Arun,Gaur Ishan,Kaur Sanmit-Loan Prediction based on machine Learning approach,IOSR Journal of computer engineering,,pp.18-21 2016.

[2] X.Frencis Jensy, V.P.Sumathi,Janani Shiva Shri-An exploratory Data Analysis for Loan Prediction based on nature of clients, International Journal of Recent Technology and Engineering (IJRTE), Volume-7 Issue-4S, November 2018

[3] Pidikiti Supriya, Myneedi Pavani, Nagarapu Saisushma,Namburi Vimala Kumari, k Vikash,-Loan Prediction by using Machine Learning Models, International Journal of Engineering and Techniques - Volume 5 Issue 2, Mar-Apr 2019

[4] Nikhil Madane, Siddharth Nanda-Loan Prediction using Decision tree,Journal of the Gujrat Research History, Volume 21 Issue 14s, December 2019

[5] Dataset Source, https://www.kaggle.com

[6] OmPrakash Yadav,Chandan Soni, Saikumar Kandakatla,Shantanu Sswanth-Loan Prediction using decision tree,International Journal of Information and computer Science, Volume 6, Issue 5, May 2019

[7] Aditi kacheria, Nidhi Shivakumar, Shreya Sawker, Archana Gupta- Loan sanctioning prediction system, International Journal of soft computing and engineering(IJSCE)- Volume-6, issue-4, september 2016

[8] Shrishti Srivastava, Ayush Garg, Arpit Sehgal, Ashok kumar – Analysis and comparison of Loan Sanction Prediction Model using Python, International journal of computer science engineering and information technology research(IJCSEITR), Vol and issue 2, June 2018

[9] Anchal Goyal, Ranpreet Kaur- A survey on ensemble model of Loan Prediction, International journal of engineering trends and application(IJETA), Vol. 3 Issue 1, Jan-Feb 2016

[10] G. Arutjothi, Dr. C. Senthamarai- Prediction of Loan Status in Commercial Bank using Machine Learning Classifier, proceedings of the International Conference on Intelligent