

A Mini Project report
on
STUDENT PLACEMENT PREDICTION

Project submitted to the Jawaharlal Nehru Technological University in
Partial fulfillment of the requirements for the award of the Degree of Bachelor of
Technology in Computer Science and Engineering

Submitted By

M. Swetha
(16891A0532)

K. Mahesh
(16891A0529)

K.V.S.Dheeraj
(16891A0522)

Under the Guidance of
Mrs.P. LAVANYA KUMARI
Associate Professor



Department of Computer Science and Engineering,
VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE, Deshmukhi
Affiliated to Jawaharlal Nehru Technological University. Hyderabad

2019-2020

VIGNAN INSTITUTE OF TECHNOLOGY AND SCIENCE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Mini project report titled **Student Placement Prediction** is being submitted by **M. Swetha**, bearing 16891A0532, **K. Mahesh Reddy**, bearing 16891A0529, **K.V.S. Dheeraj**, bearing 16891A0522 in IV BTech I semester *Computer Science & Engineering* is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Mrs. P. Lavanya Kumari
Associate Professor

Mr. G. Raja Vikram
Head of the Department

MINI PROJECT EVALUATION CERTIFICATE

This is to certify that the Mini Project work entitled “**STUDENT PLACEMENT PREDICTION**” submitted by **M. Swetha** (16-532), **K. Mahesh** (16-529), **K.V.S. Dheeraj** (16-522) has been examined and adjudged as sufficient for the partial fulfilment of the requirement of the degree of Bachelor of Technology in **Computer Science and Engineering** of Jawaharlal Nehru Technological University, Hyderabad.

External Examiner : _____

(Signature with Date)

Internal Examiner : _____

(Signature with Date)

Head of the Department : _____

(Signature with Date)

ACKNOWLEDGEMENT

Success will be crowned to people who made it reality but the people whose constant guidance and encouragement made it possible will be crowned first on the eve of success.

This acknowledgement transcends the reality of formality when we would like to express deep gratitude and respect to all those people behind the screen who guided, inspired and helped us for the completion of our project work.

We would like to thank respected Mr. G. Raja Vikram, Head of the Department for giving us such a wonderful opportunity to expand our knowledge. It helped us a lot to realize of what we study for. Secondly, we would like to thank our guide Mrs. P. Lavanya Kumari, Associate Professor for guiding us at every step and helping us know how to complete a project. Thirdly, we would like to thank our parents who patiently helped us as we went through our work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs. Next, we would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

We also extend our thanks to all the staff of the department of Computer Science and Engineering, VITS for their co-operation and support during our course work. And, we would like to thank our friends too who helped us to make our work more organized and well-stacked till the end.

Last but clearly not the least, we would thank the Almighty for giving us strength to complete our work on time.

DECLARATION

We declare that this written submission represents our ideas in our own words and where others ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

M. SWETHA (16-532)

K. MAHESH (16-529)

K.V.SAI DHEERAJ (16-522)

CONTENTS

Abstract.....	i
List of Figures.....	ii
List of Screens.....	iii
1. Introduction.....	1
1.1 Motivation.....	1
1.2 Problem definition.....	2
1.3 Objective of Project.....	2
2. EXISTING AND PROPOSED SYSTEM.....	3
2.1 Existing System and Disadvantages.....	3
2.2 Proposed System.....	3
2.3 Modules Used.....	4
2.3.1. Scikit-Learn	4
2.3.2. Training&Testing.....	5
2.3.3. Selection&Prediction.....	5
2.3.4. Numpy.....	6
2.3.5. Pandas.....	6
3. REQUIREMENTS AND ANALYSIS.....	8
3.1 Platform requirement.....	8
3.2 Software Requirement Specification.....	8
3.2.1. Software requirement.....	8
3.2.2. Hardware requirement.....	8
3.3 Flowcharts.....	9
3.3.1. System flow.....	9
3.3.2. Placement Prediction system architecture.....	10
4. DESIGN.....	11
4.1 Introduction.....	11
4.2 UML diagram.....	11
4.3 Module architecture diagram.....	12
5. IMPLEMENTATION & RESULTS.....	13
5.1 Introduction.....	13
5.2 Key functions.....	13
5.2.1. Tkinter.....	13
5.2.2. Decision Tree.....	16

5.3 Method of Implementation.....	20
5.3.1. Output Tkinter window.....	20
5.3.2. Selection.....	21
5.3.3. Rejection.....	22
6. TESTING & VALIDATION.....	23
7. CONCLUSION.....	24
7.1 Future scope.....	24
8. REFERENCES.....	25
8.1 Research paper references.....	25
8.2 Book references.....	25

ABSTRACT

Predicting the number of potential candidates is one of biggest challenges for the placement officer, since sometime a student with very good academic score doesn't get place and the one with the low academic score but good communication skills get place. Clearly, there are more than one factor influencing the placements. Finding the relation between factors can be even challenging for a trained statistician. Even companies visiting the college uses the criteria of aggregate to sort out students and sometime due to error of method they end up empty handed with no potential student. We are proposing a machine learning evaluation system which will consider many different factors like communication skills, technical skills, aggregate to determine the potential of the student. We are using Decision Tree Algorithm which gives excellent performance in classification problem with less data constraint.

LIST OF FIGURES

3.1	System flow.....	09
3.2	Student placement prediction ARCHITECTURE.....	10
4.1	UML Diagram (Activity diagram).....	11
4.2	Model architecture diagram,.....	12
5.1	Decision Tree Example.....	17

LIST OF SCREENS

5.2. Output Tkinter window.....	20
5.3. Selection.....	21
5.4. Rejection(1).....	22
5.5. Rejection(2).....	22
6.1. Testing & Validating.....	23

1.INTRODUCTION

Nowadays educational institutes are growing in high numbers. Aim of every higher educational institute is to get their students a well-paid job through their placement cell. One of the largest challenges that higher learning establishments face nowadays is to boost the placement performance of scholars. The placement prediction is additional complicated once the quality of instructional entities increases. One of the effective ways to address the challenges for improving the quality is to provide new knowledge related to the educational processes and entities to the managerial system. With the machine learning techniques, the information is often extracted from operational and historical knowledge that resides at intervals the academic organization's databases exploitation. The information set for system implementation contains data regarding past data of scholars. These knowledges square measure used for coaching the model for rule identification and for testing the model for classification. The prediction of placement status that students are most likely to achieve will help students to put in more hard work to make appropriate progress in stepping into a career in various technical fields. It will also help the teachers as well as placement cell in an institution to provide proper care towards the improvement of students in the duration of course. A high placement rate is a key entity in building the reputation of an educational institution. Hence such a system has a significant place in the educational system of any higher learning institution. We use Decision machine learning module to provide efficient and accurate results and we also provide a nice Graphical User Interface for easy interaction with the model.

1.1. MOTIVATION

- The current system generally uses only a single parameter to judge whether a student can be placed or not during the campus placements. Generally, the parameter used to judge the strengths of the student, is the academic performances during the first three years of engineering. But cracking an interview not only depends on the academic scores but also the awareness of student during the aptitude tests and interviews.
- Also, some Data Mining algorithms, while calculating the probability of a student getting selected, sometimes interpret the result having a probability of more than 100% which is not feasible and denotes a wrong interpretation to the student.

1.2. PROBLEM DEFINITION

The general Placement Prediction System considers only academic performances in order to predict whether a student can be placed or not. Judging the student based only on his academic performances would be unfair for the student, since a student could be having good aptitude, technical and communication skills but unfortunately might not be good in academic performances. It would be wrong to judge a student based only on his academic performances, since Predicting the placement of a student needs a lot of parameters to be considered. But in order to get selected in campus interview, the student must be good in technical and aptitude skills. Of course, academic performances are important but don't hold the highest importance in the outcome of student placement.

1.3. OBJECTIVE

- ☐ Time management
- ☐ Easy access
- ☐ Direct interaction with the executive

2. LITERATURE SURVEY

2.1. EXISTING SYSTEM AND DISADVANTAGES

- Previous existing system conducted a study to predict student placement status using two attributes, areas and CGPA results. They made use of K nearest neighbour, SCI-Kit leaning in machine Learning here they use only two parameters such as CGPA and arrears used algorithm takes more time for prediction not efficient.
- Also, another existing project has conducted a study to predict suitable course for the students, based on their behaviour using Neural Network Technique. TensorFlow engine includes number of intermediate node and number of deep learning layers are adjusted and compared.
- The current system generally uses only a single parameter to judge whether a student can be placed or not during the campus placements. Generally, the parameter used to judge the strengths of the student, is the academic performances during the first three years of engineering. But cracking an interview not only depends on the academic scores but also the awareness of student during the aptitude tests and interviews. Also, some Data Mining algorithms, while calculating the probability of a student getting selected, sometimes interpret the result having a probability of more than 100% which is not feasible and denotes a wrong interpretation to the student.
- Some algorithms give a negative probability which gives a wrong interpretation to the student. Judging the student only on basis of academic grades is not enough. The other parameters like aptitude and technical tests should also be taken into consideration in order to determine the outcome for the student's future.

2.2. PROPOSED SYSTEM

- Proposed system follows the Decision Tree model of classification, which removes the problems caused by previous algorithms like negative prediction and over-value prediction (more than 100% prediction value).
- Here we use Decision machine learning module to provide efficient and accurate results and we also provide a nice Graphical User Interface by Tkinter for easy interaction with the model.

2.3.MODULES USED

2.3.1. SCIKIT-LEARN

Defining scikit learn, it is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn was initially developed by David Cournapeau as a Google summer of code project in 2007. Later Matthieu Brucher joined the project and started to use it as a part of his thesis work. In 2010 INRIA got involved and the first public release (v0.1 beta) was published in late January 2010. The project now has more than 30 active contributors and has had paid sponsorship from INRIA, Google, Tinyclues and the Python Software Foundation.

In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or **features**.

Learning problems fall into a few categories:

- supervised learning, in which the data comes with additional attributes that we want to predict (Click [here](#) to go to the scikit-learn supervised learning page). This problem can be either:
 - classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of a classification problem would be handwritten digit recognition, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.

- regression: if the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.
- unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization* (Click here to go to the Scikit-Learn unsupervised learning page).

2.3.2. TRAINING AND TESTING SET

Machine learning is about learning some properties of a data set and then testing those properties against another data set. A common practice in machine learning is to evaluate an algorithm by splitting a data set into two. We call one of those sets the **training set**, on which we learn some properties; we call the other set the **testing set**, on which we test the learned properties.

2.3.3. LEARNING AND PREDICTING

In the case of the digits dataset, the task is to predict, given an image, which digit it represents. We are given samples of each of the 10 possible classes (the digits zero through nine) on which we *fit* an estimator to be able to *predict* the classes to which unseen samples belong.

In scikit-learn, an estimator for classification is a Python object that implements the methods `fit(X, y)` and `predict(T)`.

An example of an estimator is the class `sklearn.svm.SVC`, which implements support vector classification. The estimator's constructor takes as arguments the model's parameters.

```
>>> from sklearn import svm
>>> clf = svm.SVC(gamma=0.001, C=100.)
```

The `clf` (for classifier) estimator instance is first fitted to the model; that is, it must *learn* from the model. This is done by passing our training set to the `fit` method.

2.3.4. NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

2.3.5. PANDAS

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

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels

- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, **Series** (1-dimensional) and **DataFrame** (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, **DataFrame** provides everything that R's `data.frame` provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

- Easy handling of **missing data** (represented as NaN) in floating point as well as non-floating point data.
- Size mutability: columns can be **inserted and deleted** from DataFrame and higher dimensional objects.
- Automatic and explicit **data alignment**: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let *Series*, *DataFrame*, etc. automatically align the data for you in computations
- Powerful, flexible **group by** functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it **easy to convert** ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets
- Intuitive **merging** and **joining** data sets
- Flexible **reshaping** and pivoting of data sets
- **Hierarchical** labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from **flat files** (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast **HDF5 format**
- **Time series**-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

3.REQUIREMENT ANALYSIS

3.1. PLATFORM REQUIREMENT

The supported Operating Systems for client include:

- Windows XP onwards
- Linux any flavour
- Mac OS

Windows and Linux are two of the operating systems that will support comparative window. Since Linux is an open source operating system, this system which is will use in this project is developed on the Linux platform but is made compatible with windows too. The comparative window will be tested on both Linux and windows.

3.2. SOFTWARE REQUIREMENT SPECIFICATION

3.2.1. Software Requirement

The Software Requirements in this project include:

- ☐ Anaconda (spyder IDE)
- ☐ Tkinter , Pandas, numpy , modules
- ☐ Machine Learning algorithms

3.2.2. Hardware Requirement

Hardware Requirement for project development includes:

- ☐ 4 GB Ram
- ☐ 40 GB Hard DiskMinimum
- ☐ Intel Core

3.3. FLOW CHART

3.3.1.SYSTEM FLOW

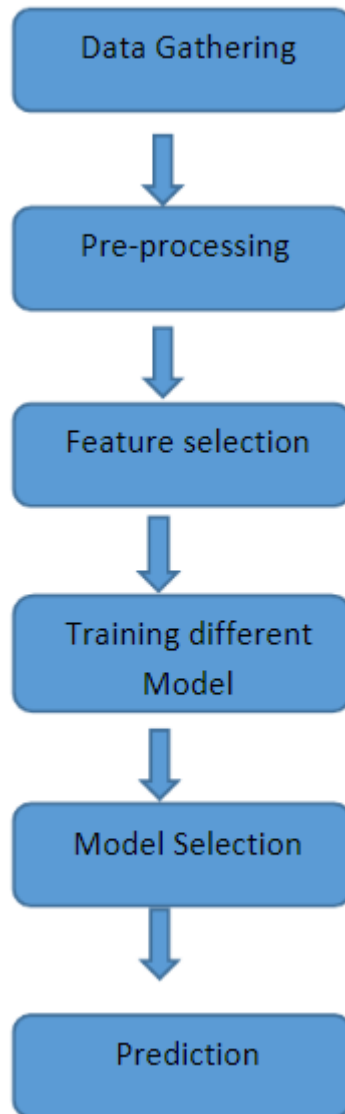


Fig.3.1

Description

The Data was collected from Training and placement department of VIGNAN HYD which consist of all the students of Bachelor of Engineering (B.E) from different branches of their campus.

- We have dropped most of the NA values which we have found in our data, since some of the students are detained. And in some cases we have replaced the NA values.
- Using LabelEncoder from preprocessing API in sklearn encoded the labels of columns.
- The whole process is done in data gathering and preprocessing.

3.3.2. PLACEMENT PREDICTION SYSTEM ARCHITECTURE

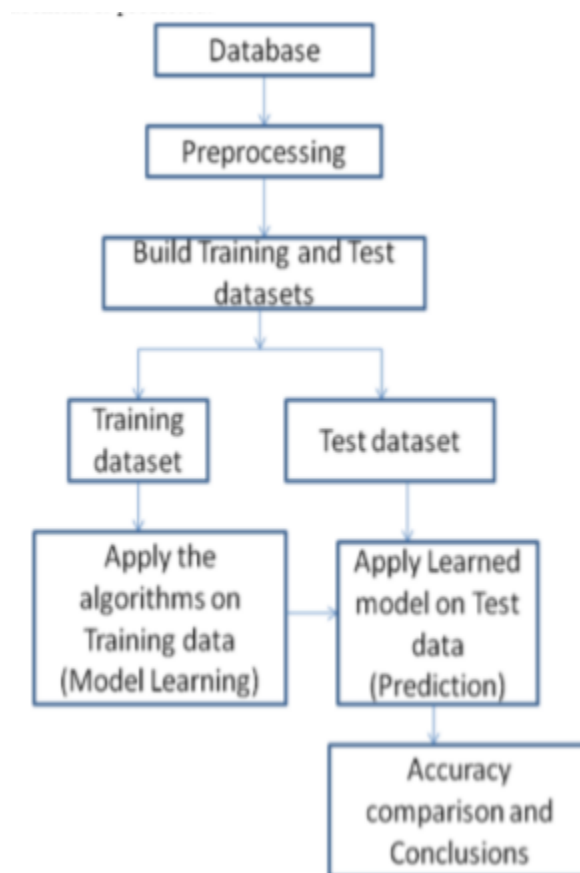


Fig.3.2

4.DESIGN

4.1. INTRODUCTION

Design is the most important step in the development phase. Once the software requirements have been analyzed and specified the software design involves three technical activities design, coding, implementation and testing that are required to build and verify the product. The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer requirements into finished software or a system. Design is the place where quality is fostered in development. Software design is a process through which requirements are translated in to a representation of software.

4.2. UML DIAGRAM (Activity diagram)

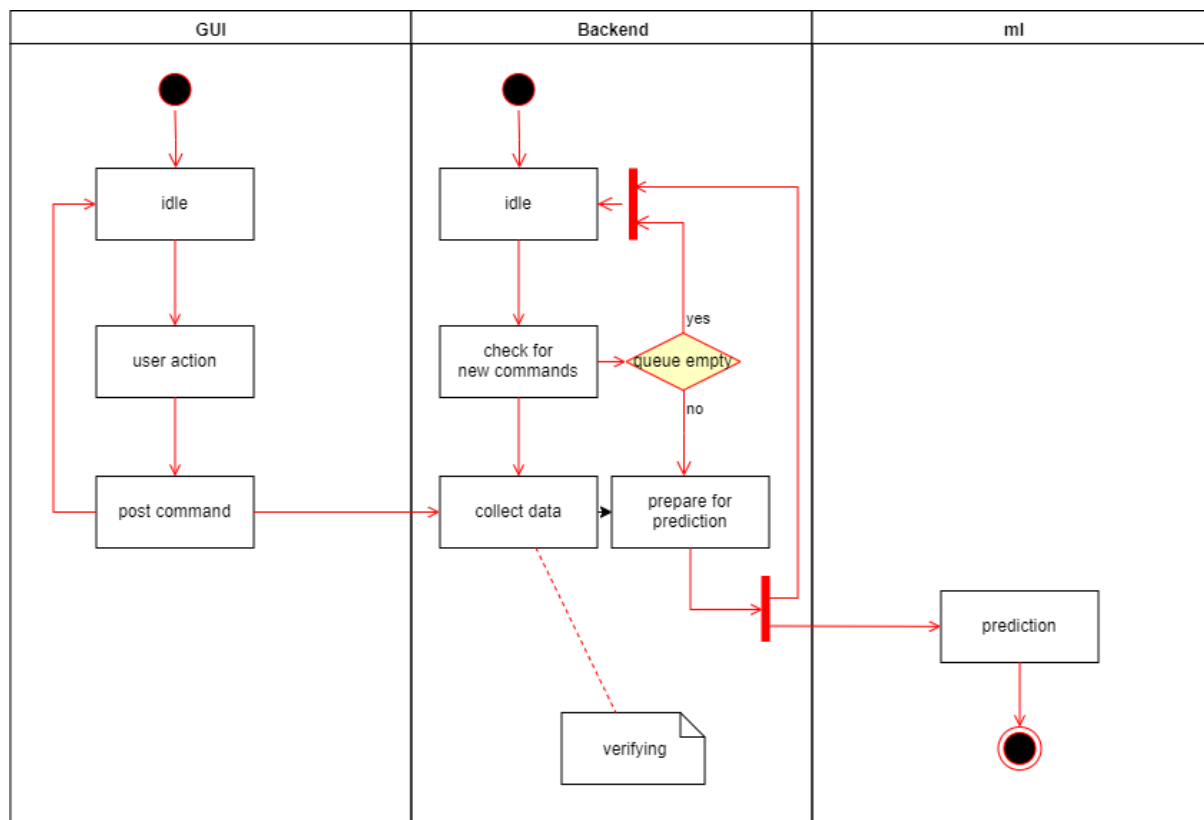


Fig. 4.1

4.3. MODEL ARCHITECTURE DIAGRAM

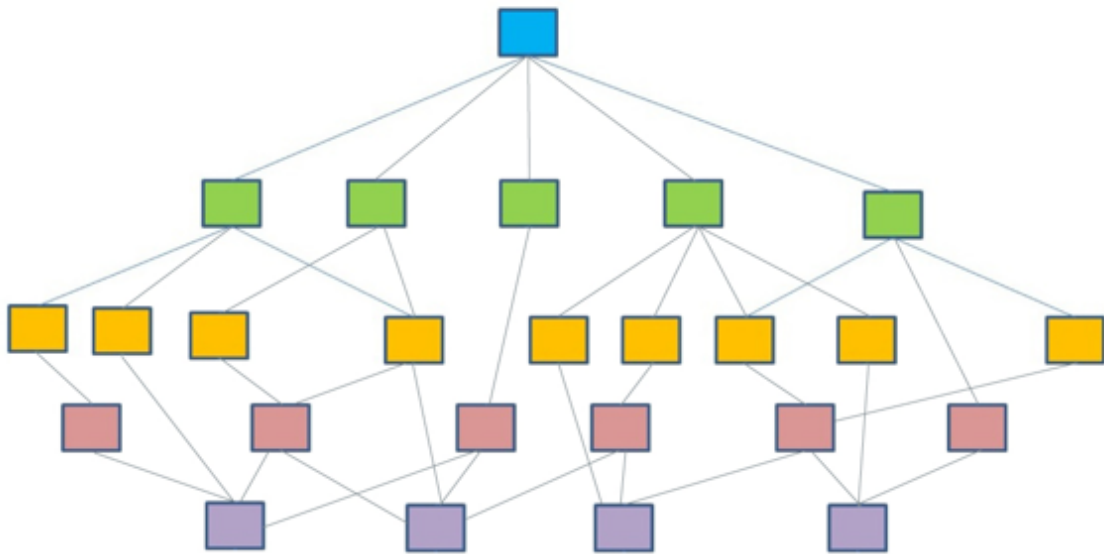


Fig. 4.2

5. IMPLEMENTATION AND RESULTS

5.1 INTRODUCTION

The implementation phase, the project plan is put into motion and the work of the project is performed. It is important to maintain control and communicate as needed during implementation. Progress is continuously monitored and appropriate adjustments are made and recorded as variances from the original plan.

5.2 KEY FUNCTION

5.2.1Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Python 2.7 and Python 3.1 incorporate the "themed Tk" ("ttk") functionality of Tk 8.5. This allows Tk widgets to be easily themed to look like the native desktop environment in which the application is running, thereby addressing a long-standing criticism of Tk (and hence of Tkinter). There are several popular GUI library alternatives available, such as wxPython, PyQt (PySide), Pygame, Pyglet, and PyGTK.

Window

This term has different meanings in different contexts, but in general it refers to a rectangular area somewhere on the user's display screen.

Top Level Window

A window that exists independently on the screen. It will be decorated with the standard frame and controls for the desktop manager. It can be moved around the desktop, and can usually be resized.

Widget

The generic term for any of the building blocks that make up an application in a graphical

user interface.

- Core widgets: The containers: frame, toplevel, paned window. The buttons: button, radiobutton, checkbutton (checkbox), menubutton (combobox). The text widgets: label, labelframe, message, text. The entry widgets: scale, scroll, listbox, slider, spinbox, entry (singleline), text (multiline), and canvas (vector and pixel graphics).
- There are the extension widgets: tk_optionMenu, tk_dialog, tk_messageBox, tk_getOpenFile, tk_getSaveFile, tk_chooseColor, tk_chooseDirectory.
- **The ttk widgets:** The ttk widgets coexist with other widgets which they can replace.^[7] There are ttk::button, ttk::checkbutton, ttk::combobox, ttk::entry, ttk::frame, ttk::label, ttk::labelframe, ttk::menubutton, ttk::notebook, ttk::panedwindow, ttk::progressbar, ttk::radiobutton, ttk::scale, ttk::scrollbar, ttk::separator, ttk::sizegrip, ttk::spinbox, ttk::treeview.

Frame

In Tkinter, the Frame widget is the basic unit of organization for complex layouts. A frame is a rectangular area that can contain other widgets.

Child and parent

When any widget is created, a parent-child relationship is created. For example, if you place a text label inside a frame, the frame is the parent of the label.

A minimal application

Here is a minimal Python 3 Tkinter application with one widget: (For Python 2, the only difference is the word "tkinter" in the import command will be capitalized to "Tkinter.")

```
1 #!/usr/bin/env python3
2 from tkinter import *
3 root = Tk()                                # Create the root (base) window
4 w = Label(root, text="Hello, world!")      # Create a label with words
5 w.pack()                                  # Put the label into the window
6 root.mainloop()                           # Start the event loop
```

Process

There are four stages to creating a widget

Create

create it within a frame

Configure

change the widgets attributes

Pack

pack it into position so it becomes visible

Bind

bind it to a function or event.

These are often compressed and the order can vary.

Simple application

Using the object orientated paradigm in Python, a simple program would be:

```
1 #!/usr/bin/env python3
2 import tkinter as tk
3
4 class Application(tk.Frame):
5
6     def __init__(self, master=None):
7         super(Application, self).__init__(master)
8         self.grid()
9         self.createWidgets()
10
11     def createWidgets(self):
12         self.mondialLabel = tk.Label(self, text='Hello World')
13         self.mondialLabel.config(bg="#00ffff")
14         self.mondialLabel.grid()
15         self.quitButton = tk.Button(self, text='Quit', command=self.quit)
16         self.quitButton.grid()
17
18 app = Application()
19 app.master.title('Sample application')
20 app.mainloop()
```

- line 1: Hashbang directive to the program launcher, allowing the selection of an appropriate interpreter executable, when self-executing.

- line 2: This line imports the tkinter module into your program's namespace, but renames it as tk.
- line 4: The application class inherits from Tkinter's Frame class.
- line 6: Calls the constructor for the parent class, Frame.
- line 7: This is necessary to make the application actually appear on the screen.
- line 11: Defining the widgets
- line 12: Creates a label, named MondialLabel with the text "Hello World"
- line 13: Sets the MondialLabel background colour to cyan
- line 14: Places the label on the application so it is visible using the grid geometry manager method
- line 15: Creates a button labeled “Quit”.
- line 16: Places the button on the application. Grid, place and pack are all methods of making the widget visible
- line 18: The main program starts here by instantiating the Application class.
- line 19: This method call sets the title of the window to “Sample application”.
- line 20: Starts the application's main loop, waiting for mouse and keyboard events.

5.2.2. DECISION TREE

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning.

How can an algorithm be represented as a tree?

For this let's consider a very basic example that uses titanic data set for predicting whether a passenger will survive or not. Below model uses 3 features/attributes/columns from the data set, namely sex, age and sibsp (number of spouses or children along).

A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/**internal node**, based on which the tree splits into branches/ **edges**. The end of the branch that doesn't split anymore is the decision/**leaf**, in this case, whether the passenger died or survived, represented as red and green text respectively.

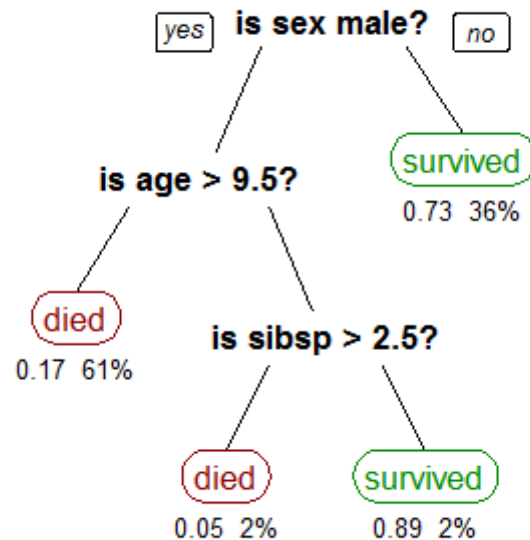


Fig.5.1

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The **feature importance** is clear and relations can be viewed easily. This methodology is more commonly known as **learning decision tree from data** and above tree is called **Classification tree** as the target is to classify passenger as survived or died. **Regression trees** are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

So, what is actually going on in the background? Growing a tree involves deciding on **which features to choose** and **what conditions to use** for splitting, along with knowing when to stop. As a tree generally grows arbitrarily, **you will need to trim it down** for it to look beautiful. Lets start with a common technique used for splitting.

Recursive binary splitting:

In this procedure all the features are considered and different split points are tried and tested using a cost function. The split with the best cost (or lowest cost) is selected.

Consider the earlier example of tree learned from titanic dataset. In the first split or the root, all attributes/features are considered and the training data is divided into groups based on this split. We have 3 features, so will have 3 candidate splits. Now we will **calculate how much accuracy each split will cost us, using a function. The split that costs least is chosen**, which in our example is sex of the passenger. This **algorithm is recursive in nature** as the groups formed can be sub-divided using same strategy. Due to this procedure, this algorithm is also known as the **greedy algorithm**, as we have an excessive desire of lowering the cost. **This makes the root node as best predictor/classifier.**

Cost of a split

Lets take a closer look at **cost functions used for classification and regression**. In both cases the cost functions try to **find most homogeneous branches, or branches having groups with similar responses**. This makes sense we can be more sure that a test data input will follow a certain path.

Regression : $\text{sum}(y - \text{prediction})^2$

Lets say, we are predicting the price of houses. Now the decision tree will start splitting by considering each feature in training data. The mean of responses of the training data inputs of particular group is considered as prediction for that group. The above function is applied to all data points and cost is calculated for all candidate splits. Again the split with lowest cost is chosen. Another cost function involves reduction of standard deviation, more about it can be found [here](#).

Classification : $G = \text{sum}(pk * (1 - pk))$

A Gini score gives an idea of how good a split is by how mixed the response classes are in the groups created by the split. Here, pk is proportion of same class inputs present in a particular group. A perfect class purity occurs when a group contains all inputs from the same class, in which case pk is either 1 or 0 and $G = 0$, where as a node having a 50–50 split of classes in a group has the worst purity, so for a binary classification it will have $pk = 0.5$ and $G = 0.5$.

When to stop splitting?

You might ask **when to stop growing a tree?** As a problem usually has a large set of features, it results in large number of split, which in turn gives a huge tree. Such trees are complex and can lead to overfitting. So, we need to know when to stop? One way of doing this is to **set a minimum number of training inputs to use on each leaf**. For example we can use a minimum of 10 passengers to reach a decision(died or survived), and ignore any leaf that takes less than 10 passengers. Another way is to set **maximum depth** of your model. **Maximum depth refers to the**

the length of the longest path from a root to a leaf.

Pruning

The performance of a tree can be further increased by **pruning**. It involves **removing the branches that make use of features having low importance**. This way, we reduce the complexity of tree, and thus increasing its predictive power by reducing overfitting.

Pruning can start at either root or the leaves. The simplest method of pruning starts at leaves and removes each node with most popular class in that leaf, this change is kept if it doesn't deteriorate accuracy. Its also called **reduced error pruning**. More sophisticated pruning methods can be used such as **cost complexity pruning** where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree. This is also known as **weakest link pruning**.

Advantages of CART

- Simple to understand, interpret, visualize.
- Decision trees implicitly perform variable screening or feature selection.
- Can handle both numerical and categorical data. Can also handle multi-output problems.
- Decision trees require relatively little effort from users for data preparation.
- Nonlinear relationships between parameters do not affect tree performance.

Disadvantages of CART

- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This is called variance, which needs to be lowered by methods like bagging and **boosting**.
- Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.

- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the data set prior to fitting with the decision tree.



5.3. IMPLEMENTATION

5.3.1. OUTPUT TKINTER WINDOW:

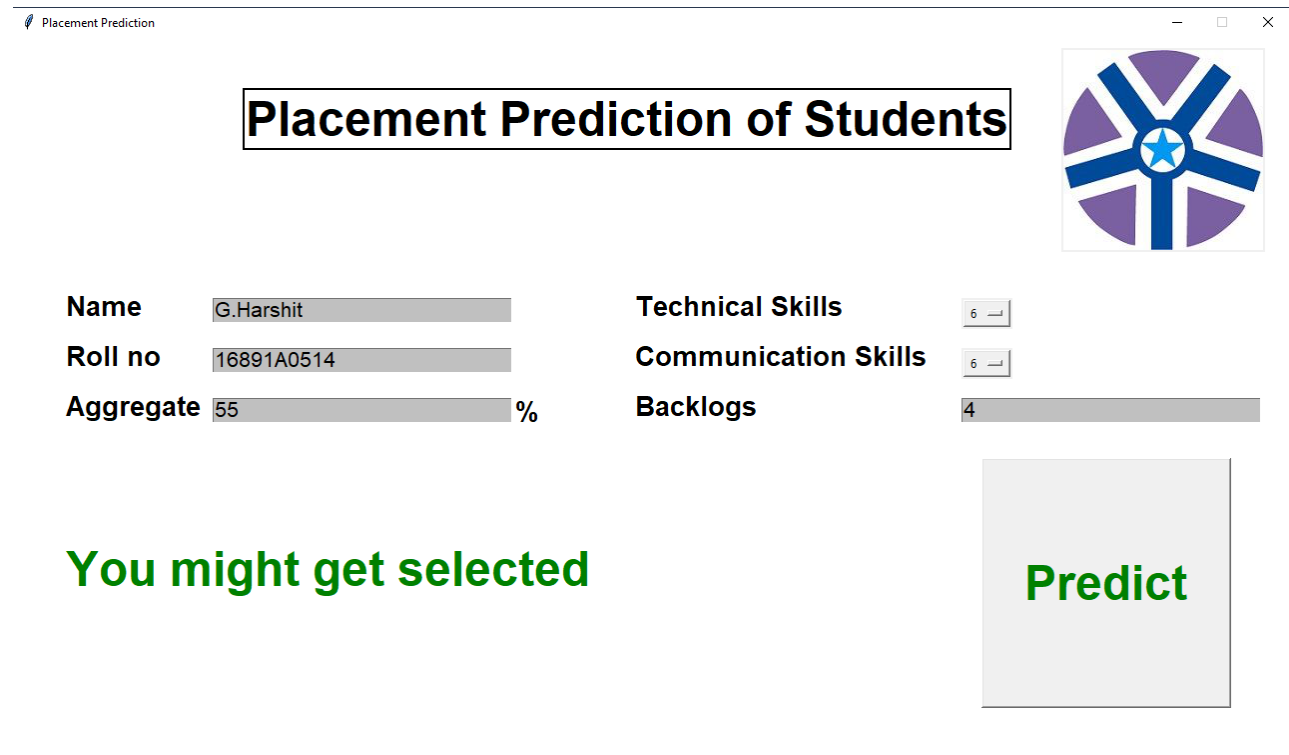
The screenshot shows a Tkinter window titled "Placement Prediction". At the top center, there is a large text box containing the title "Placement Prediction of Students". To the right of this text box is a circular logo with a blue star in the center and purple and blue segments. Below the title, there are input fields for "Name", "Roll no", and "Aggregate %". To the right of these fields are labels for "Technical Skills", "Communication Skills", and "Backlogs". The "Technical Skills" and "Communication Skills" labels are followed by dropdown menus, both currently showing "None". The "Backlogs" label is followed by a text input field. At the bottom right, there is a large button labeled "Predict" in green text.

Fig 5.2

This is the output window through which we can make predictions. Here Roll no and Name does't matter as they will be unique for every student. But we have to give values for remaining

fields as they are mandatory and based on them the whole model is formed.

5.3.2. SELECTION



Placement Prediction of Students

Name **Technical Skills**

Roll no **Communication Skills**

Aggregate % **Backlogs**

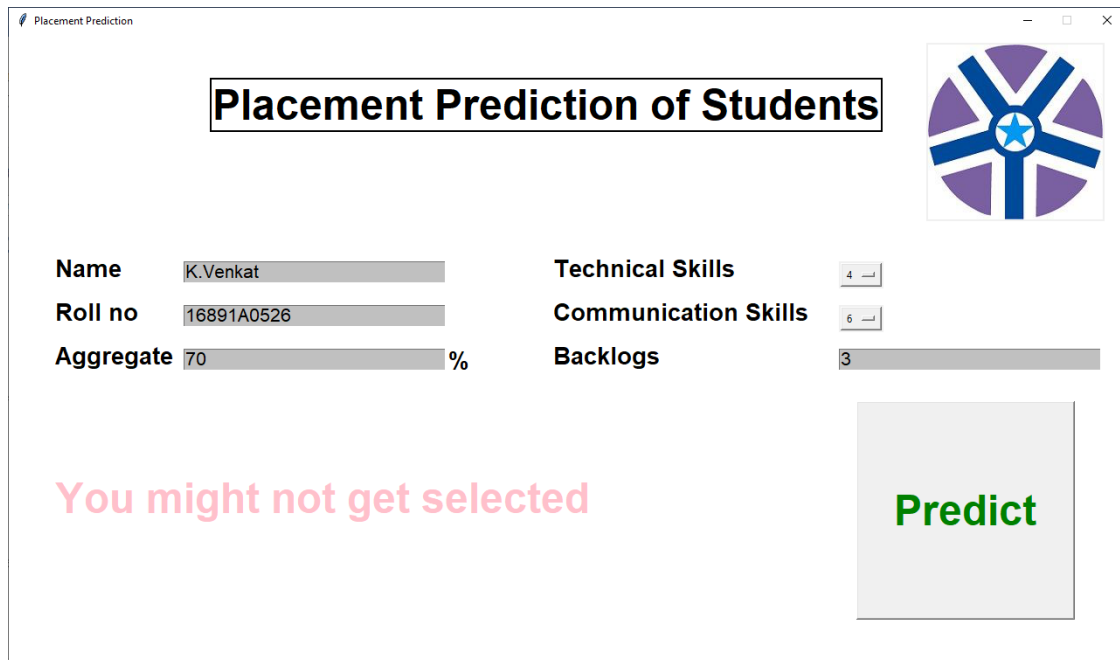
You might get selected

Predict

Fig 5.3

- Here [FIG 5.2] we can see that even if the student has low CGPA and contains more backlogs , he has a chance of getting placed . As here the algorithm has taken the data and formed a model, which also considers technical skills and communication skills.
(He has low CGPA and more backlogs but he has good knowledge on technical skills and has well communication skills too, based on that he has a chance of getting placed.
- But if the student has more percentage of CGPA as well as less backlogs , with a low technical skills and well communication then there is a chance of student get rejected by the company.
- These predictions are formed based on a model which the algorithm has learned from the previous year's data, where the student got placed or not.

5.3.3. REJECTION



Placement Prediction of Students

Name: K.Venkat

Roll no: 16891A0526

Aggregate: 70 %

Technical Skills: 4

Communication Skills: 6

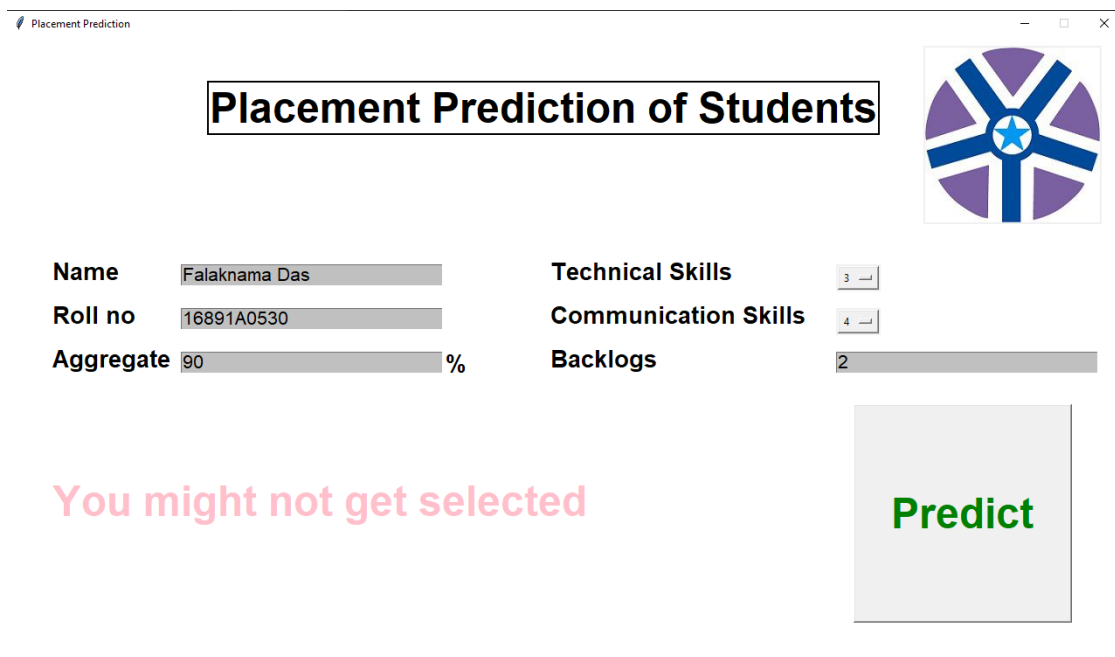
Backlogs: 3

You might not get selected

Predict

Fig 5.4

- According to the target function which the algorithm has formed and came to a conclusion for predicting whether a student will get placed or not.
- Even if the student has CGPA upto 90 % if he is poor in technical skills , model predicts its as 'REJECTION'.



Placement Prediction of Students

Name: Falaknama Das

Roll no: 16891A0530

Aggregate: 90 %

Technical Skills: 3

Communication Skills: 4

Backlogs: 2

You might not get selected

Predict

Fig. 5.5

6.TESTING AND VALIDATING

In Tkinter window, users will have to enter their details which are required to fill to know whether they will get placed or not.

Fig.6.1

Other than name and roll no every other field needs to be filled , which contains a criteria.

By entering the correct values one can know the prediction whether he can be placed or not.

7. CONCLUSION

Student Placement Prediction is a system which predicts student placement status using machine learning techniques. Many projects are there related to educational sector, all these mainly concentrate on student performance predictions. All these predictions help the institute to improvise the student performance and can come up with 100% results. Many of the previous projects concentrate on a less number of parameters such as CGPA and Arrears for placement status prediction which leads to less accurate results, but proposed work contains many educational parameters to predict placement status which will be more accurate.

7.1. FUTURE SCOPE

- Employing such a system is a proactive way to use data to manage, operate, and evaluate educational institute in a better way. Depending on the quality and implementation of the underlying data, such a system could address a wide range of problems by distilling data from any combination of education records maintenance system. It helps an educational institute improving the quality and placements of students being graduated from their institute. This system can also be implemented in different non educational institutes like business corporates, sports academies and manufacturing companies where the challenge would be taking into consideration the current market scenario as one of the most important factors affecting quality of their products and employee.
- There are many optimization algorithms available which can reduce the number of loops required for higher stability. One of them is known as Stochastic Gradient Decent Algorithm. Newton's gradient method and hessian matrix methods can also give you the good value.

8.REFERENCES

8.1. RESEARCH PAPER REFERENCES:

- [1] “Student Placement Analyzer: A Recommendation System Using Machine Learning” 2017 International Conference on advanced computing and communication systems (ICACCS-2017), Jan 06-07,2017, Coimbatore, INDIA.
- [2] “Prediction Model for Students Future Development by Deep Learning and TensorFlow Artificial Intelligence Engine” 2018 4th IEEE International Conference on Information Management.

8.2. BOOK REFERENCES:

- [1] Master Machine Learning Algorithms by JASON BROWN LEE
- [2] Hands on machine learning with sci-kit learn 2017 by Aurélien Géron

