"Student Performance Prediction System"

Prepared by
Priya Nasit(16IT057)
Dhvani Raval (16IT106)

Under the supervision of

Dr. Amit Thakkar

Prof. Purvi Prajapati

Prof. Chadni Shah

Prof. Jay Patel

A Report Submitted to

Charotar University of Science and Technology

for Partial Fulfillment of the Requirements for the

Degree of Bachelor of Technology

in Information Technology

IT350 Software Group Project-III (6th sem)

Submitted at

DEPARTMENT OF INFORMATION TECHNOLOGY

Chandubhai S. Patel Institute of Technology

At: Changa, Dist: Anand – 388421

April 2019

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

This is to certify that the report entitled "**Student Performance Prediction System**" is a bonafied work carried out by Priya Nasit, Dhvani Raval under the guidance and supervision of **Dr. Amit Thakkar, Prof. Purvi Prajapati, Prof. Jay Patel , Prof. Chadni Shah** for the subject **Software Group Project-III(IT350)** of **6th** Semester of Bachelor of Technology in **Information Technology** at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate **herself**, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,

Dr.Amit Thakkar
Associate Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Purvi Prajapati
Assistant Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Jay Patel
Assistant Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Chandani Shah
Assistant Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Parth Shah
Head & Associate Professor
Department of Information Technology
CSPIT, Changa, Gujarat.

ACKNOWLEDGEMENT

My sincere thanks to our internal project guide **Dr. Amit Thakkar, Prof. Purvi Prajapati, Prof. Chadni Shah, Prof. Jay Patel** for giving us valuable inputs and ideas right from the selection of topic for project till its successful completion. Without his constant encouragement we could not have been able to achieve what we have.

I thank **Dr. Parth Shah(HOD IT department)** for his ongoing support and encouragement in every aspect. Last but not the least, entire staff of Department of IT Engineering for guiding me thoughts and vision. The successful completion of my project would not have been possible without the dedicated support from all our mentors, family and friends.

It's my good fortune that we had support and well wishes of many. We are thankful to all and those names which have been forgotten to acknowledge here but contributions have not gone unnoticed.

With Sincere Regards,
Priya Nasit(16IT057)
Dhvani Raval(16IT106)

ABSTRACT

Student performance prediction is an area of concern for educational institutions.Machine Learning plays an important role in the business world and it helps to the educational institution to predict and make decisions related to the students' academic status. The existing system is a system which maintains the student information in the form of numerical values and it just stores and retrieve the information what it contains. So the system has no intelligence to analyze the data. The prediction with high accuracy in student's performance in beneficial as it helps in identifying the student with low academic achievements at the early stage of academics.

In this work, classifier techniques based on six representative learning algorithms, namely Naive Bayes, Logistic Regression, k-Nearest Neighbour, Support Vector Machine, Decision Tree classification, Random Forest classification and ANN. These six learning methods have been compared separately with respect to the training and test sets. Random Forest Classification is found to be the best classifier for predicting the student's result based on the marks obtained in the semester, Permanent City, Gender, Actual Cast Category,12^{th} Percentage, Board. We would also discuss how these machine learning models can help to improve an education system by considering the different factors in terms of accuracy, Specificity, Precision, Prevalence, Recall, F-Measure in results.

Keywords and terms: student performance, machine learning, naïve Bayes classification, Logistic Regression, k-Nearest Neighbour, Support Vector Machine, Decision Tree classification, Random Forest classification and ANN(Artificial Neural Network).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter:1 Introduction

1.1 Project Overview

Student performance prediction system - proposal is efficient approach for classifying student performance based on data. Prediction of student academic performance helps instructors develop a good understanding of how well or how poorly the students in their classes will perform, so instructors can take proactive measures to improve student learning. A major benefit of machine learning is its ability to predict student performance. By "learning" about each student, the technology can identify weaknesses and suggests ways to improve, such as additional practice tests.

In this we examines the application of machine learning algorithms to predict whether a student will have distinction, first class or second class. The specific focus of the thesis is the comparison of machine learning methods and feature engineering techniques in terms of how much they improve the prediction performance. Feature engineering, the process of modification and selection of the features of a data set, was used to improve predictions made by these learning algorithms.

In this, prediction of students' academic performance from educational database particularly using their scores, with no economic, social and psychological factors. Such systems are beneficial to the students in improving performance of a student. The purpose of continuous evaluation system is to help regular students.

1.1.1 Machine Learning

With the wide usage of computers and internet, there has recently been a huge increase in publicly available data that can be analyzed. Be it online sales information, website traffic, or user habits, data is generated everyday. Such a large amount of data present both a problem and an opportunity. The problem is that it is difficult for humans to analyze such large data. The opportunity is that this type of data is ideal for computers to process, because it is stored digitally in a well-formatted way, and computers can process data much faster than humans.

The concept of machine learning is something born out of this environment. Computers can analyze digital data to find patterns and laws in ways that is too complex for a human to do. The basic idea of machine learning is that a computer can automatically learn from experience (Mitchell, 1997). Although machine learning applications vary, its general function is similar throughout its applications. The computer analyzes a large amount of data, and finds patterns and rules hidden in the data. These patterns and rules are mathematical in nature, and they can be easily defined and processed by a computer. The computer can then use those rules to meaningfully characterize new data. The creation of rules from data is an automatic process, and it is something that continuously improves with newly presented data.

Evaluation Matrices can be used in evaluating the predictive models. Algorithms were compared to each other in terms indicator values, to determine which algorithm provides the best results. In addition to the algorithm choice, the importance of feature engineering was also tested. To improve the prediction performance, the data sets were modified by variable selection and custom variable creation. Finally, improvements made by feature engineering were compared to improvements made by algorithm choice, to see if one is a more determinant factor than the other. Results of the comparison indicates that feature engineering provides better improvements than method selection.

1.2 Scope

Educational organizations are one of the important parts of our society and playing a vital role for growth and development of any nation. Information like marks obtained in the semester, Permanent City, Gender, Actual Cast Category,12th Percentage, Board were collected from the students management system, to predict the performance at the end of the semester. The faculty cannot find out students abilities and their interest easily so that they can enhance them in it. Thus it may affect with university results, placement and career of individual. The impact is- it help us from fulfilling mission and vision of the institute.

1.3 Objective

The main objective of higher education institution is to present quality education to its students. One way to accomplish the higher level of quality in higher education scheme is by predicting student's academic performance and thereby taking early actions to improve student's performance and teaching quality.

1.4 Problem and Solution Statement

For building student performance prediction system we developed the different classification algorithms and compare their performances with each other.

Chapter:2 System Analysis

2.1 Tools and Technology

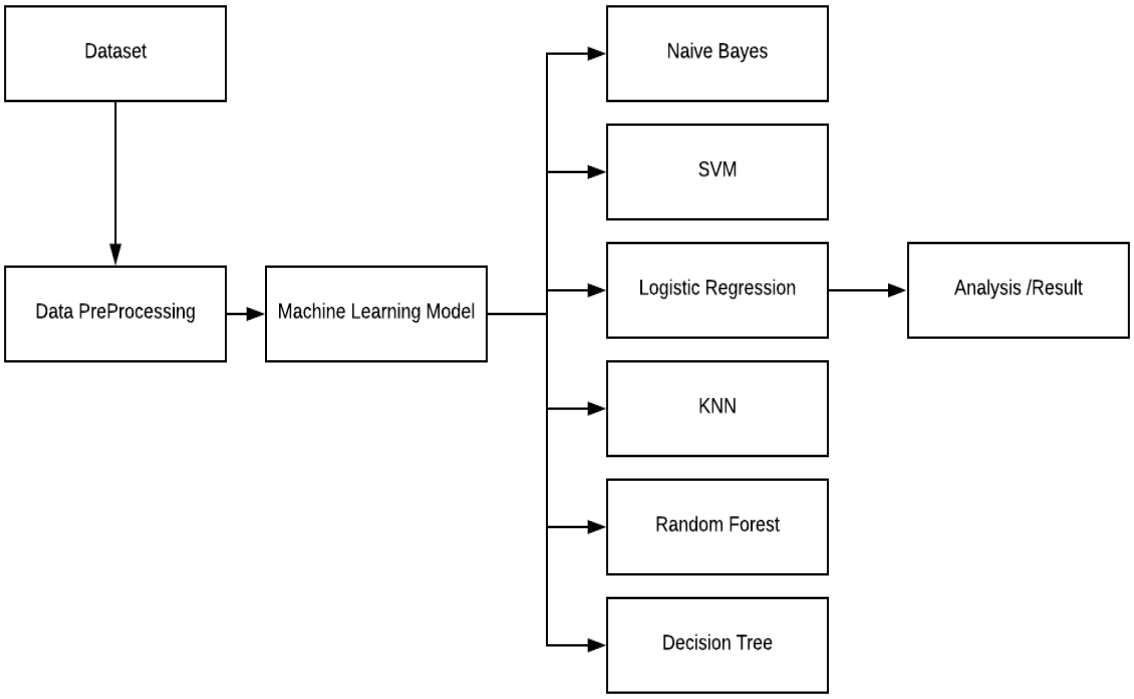
2.2.1 Software requirement

1. Operating system : Microsoft windows
2. Tool: Anaconda Navigator
3. Algorithms:
    - Naive Bayes
    - Logistic Regression
    - k-Nearest Neighbour
    - Support Vector Machine
    - Decision Tree classification
    - Random Forest classification
    - ANN
4. Technology : Machine Learning

2.2.2 Programming Language

1. Python

Chapter:3 System Design

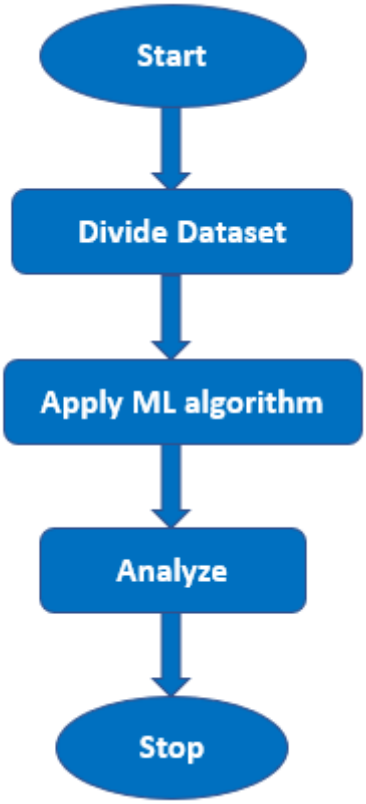


3.1 System Design

Chapter:4 Implementation

4.1 Methodology:

The main aim of the system is to predict the future performance of the student using certain data of the student such as pervious semester marks,12th percentage etc. After predicting the student performance, the system will also compare the results generated by classification algorithms and there after determine which of them is more accurate and efficient.



4.1.1 Work Flow

4.2 Algorithms

4.2.1 Logistic Regression

Logistic Regression is used when the dependent variable(target) is categorical.

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.

Logistic regression can be expressed as:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

4.2.2 Naïve Bayes Classifier

Naïve Bayes classification is a machine learning method relying on the Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

where A and B are two different events, P(A) and P(B) are the probability of A and B occurring, respectively. P(A|B) is the probability of A occurring given that B has already occurred.

4.2.3 K Nearest Neighbor

The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labelled points and uses them to learn how to label other points. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point.

4.2.4 Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new examples.

4.2.5 Decision Tree Classification

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated

decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

4.2.6 Random Forest Classification

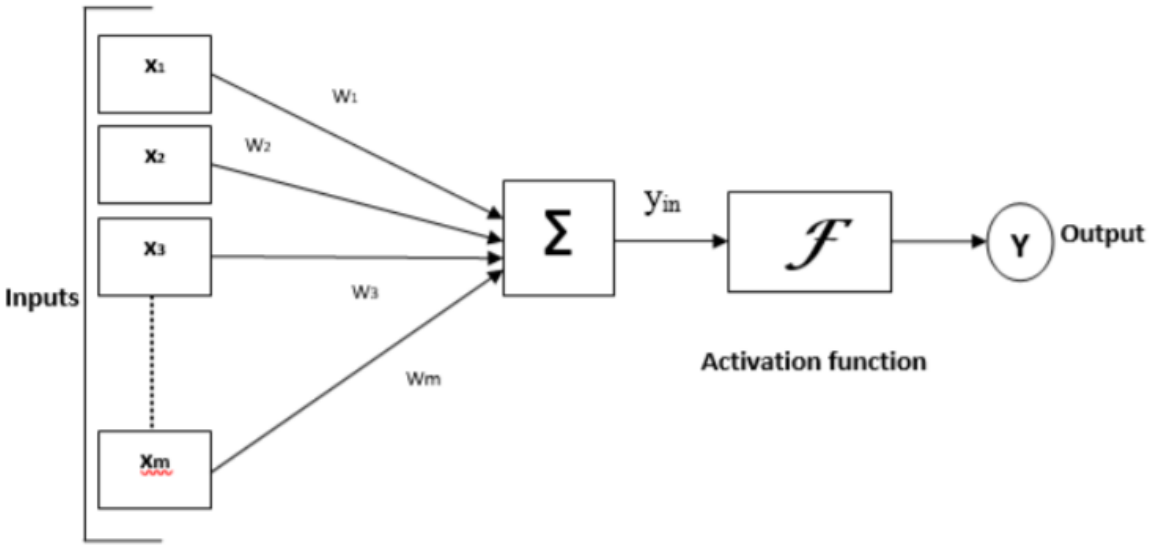
Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks. In this post, you are going to learn, how the random forest algorithm works and several other important things about it.

4.2.7 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks. ANNs are also named as "artificial neural systems," or "parallel distributed processing systems," or "connectionist systems." ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units. These units, also referred to as nodes or neurons, are simple processors which operate in parallel.

Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal. This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

4.2.7.1 Model Of ANN



4.2.7.1Model Of ANN

4.3 Feature Engineering

In machine learning, feature engineering is the process of selecting or creating features (variables) in a data set to improve machine learning results. Feature selection can include removing unnecessary or redundant features. The process of removing unnecessary variables requires assessing the relevance of the variable. This can be done by creating a model to test the correlation of the variable with the dependent variable. Feature creation includes modifying the variables and creating new ones by combining multiple different variables.

The first use of feature engineering in the thesis is the selection of the relevant variables. Input data may contain too many variables, some of which do not improve the prediction performance, and thus make the predictive model overly complicated. In such a case, unnecessary variables must be removed to make the model more efficient. Deciding which variable to remove can be done manually using domain knowledge or it can be done automatically. In the case of this, feature selection was done by observing the output of the all regression model to find how much correlation each variable has with the dependent variable.

4.3.1 Features Correlation

	Sr No	Sem-1	Sem-2	Sem-3	Sem-4	\
Sr No	1.000000	-0.035273	0.040603	0.038311	0.060860	
Sem-1	-0.035273	1.000000	0.791965	0.722224	0.668897	
Sem-2	0.040603	0.791965	1.000000	0.788797	0.765746	
Sem-3	0.038311	0.722224	0.788797	1.000000	0.846746	
Sem-4	0.060860	0.668897	0.765746	0.846746	1.000000	
Sem-5	0.078033	0.647873	0.728468	0.772562	0.827784	
CGPA	0.043281	0.724112	0.780898	0.803608	0.813682	
Gender	-0.003397	0.053619	0.118604	0.130571	0.192898	
Actual Cast Category	-0.017767	-0.083969	-0.118388	-0.107417	-0.134451	
Permanent City	-0.108376	-0.007193	-0.044955	0.000793	-0.019080	
12th Marks	0.012356	0.560311	0.546707	0.474537	0.458354	
Board	-0.000135	0.099937	0.111245	0.088840	0.083309	
Performance	-0.085614	-0.622992	-0.665813	-0.690656	-0.706548	

	Sem-5	CGPA	Gender	Actual Cast Category
Sr No	0.078033	0.043281	-0.003397	-0.017767
Sem-1	0.647873	0.724112	0.053619	-0.083969
Sem-2	0.728468	0.780898	0.118604	-0.118388
Sem-3	0.772562	0.803608	0.130571	-0.107417
Sem-4	0.827784	0.813682	0.192898	-0.134451
Sem-5	1.000000	0.834903	0.162867	-0.130128
CGPA	0.834903	1.000000	0.140407	-0.134478
Gender	0.162867	0.140407	1.000000	-0.050556
Actual Cast Category	-0.130128	-0.134478	-0.050556	1.000000
Permanent City	-0.015853	0.003270	-0.027506	0.099252
12th Marks	0.432598	0.502627	0.013075	-0.131882
Board	0.076351	0.091851	0.071748	-0.072997
Performance	-0.720083	-0.845978	-0.097948	0.096860

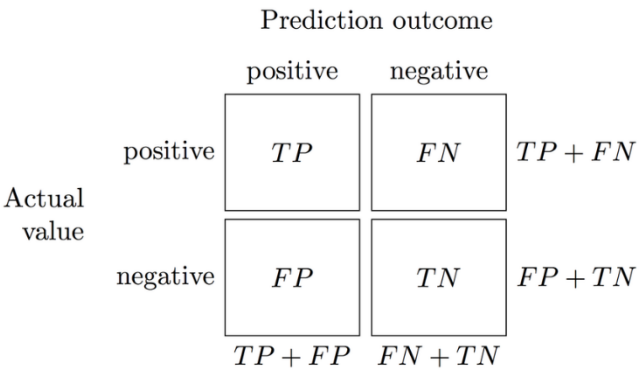
	Permanent City	12th Marks	Board	Performance
Sr No	-0.108376	0.012356	-0.000135	-0.085614
Sem-1	-0.007193	0.560311	0.099937	-0.622992
Sem-2	-0.044955	0.546707	0.111245	-0.665813
Sem-3	0.000793	0.474537	0.088840	-0.690656
Sem-4	-0.019080	0.458354	0.083309	-0.706548
Sem-5	-0.015853	0.432598	0.076351	-0.720083
CGPA	0.003270	0.502627	0.091851	-0.845978
Gender	-0.027506	0.013075	0.071748	-0.097948
Actual Cast Category	0.099252	-0.131882	-0.072997	0.096860
Permanent City	1.000000	0.038529	-0.038460	0.008203
12th Marks	0.038529	1.000000	0.041440	-0.454476
Board	-0.038460	0.041440	1.000000	-0.043050
Performance	0.008203	-0.454476	-0.043050	1.000000

4.3.1 Features Corelation

4.4 Evaluation Methods

In order to evaluate the effectiveness of a prediction model, predicted values must be compared with actual values. There are multiple criteria for prediction effectiveness.

Confusion Matrix(two class classification)



4.4.1Two Class Confusion Matrix

Measure	Formula	Evaluation Focus
Accuracy	$\frac{TP+TN}{(TP+FN+FP+TN)}$	Overall Effectiveness of Classifier.
Precision	$\frac{TP}{TP+FP}$	Class Agreement of the data lebels with the positive labels given by the classifier.
Recall (Sensitivity)	$\frac{TP}{TP+FN}$	Effectiveness of a classfier to identify positive labels.
Specificity	$\frac{TN}{FP+TN}$	How effectively a classfier identifies negative labels.
F_1 Score	$\frac{2 * precision * recall}{precision + recall}$	The **F_1score** is the harmonic average of the precision and recall,where **F_1score** reaches its best value at 1 (perfect precision and recall) and worst at 0.

4.4.2 Calculation Table

Three Class Confusion Matrix

	A	B	C	
A	T_A	F_{AB}	F_{AC}	$FN_A = F_{AB}+F_{AC}$
B	F_{BA}	T_B	F_{BC}	$FN_B = F_{BA}+F_{BC}$
C	F_{CA}	F_{CB}	T_C	$FN_C = F_{CA}+F_{CB}$
	$FP_A=F_{BA}+F_{CA}$	$FP_B=F_{AB}+F_{CB}$	$FP_C= F_{Ac}+ F_{BC}$	

4.4.3 Three Class Confusion Matrix

4.5 Dataset Features

Attribute Name	Value	Description
Sem1 SGPA	0<=sem1_marks<=10	Sem1 Result
Sem2 SGPA	0<=sem2_marks<=10	Sem2 Result
Sem3 SGPA	0<=sem3_marks<=10	Sem3 Result
Sem4 SGPA	0<=sem4_marks<=10	Sem4 Result
Sem5 SGPA	0<=sem5_marks<=10	Sem5 Result
CGPA	0<=CGPA<=10	Average upto 5 Semester
Gender	{Male, Female}	Male=0 Female=1
Cast	{OPEN,OBC,ST,SC,SEBC}	OPEN=0 OBC=1 SC=2 ST=3 SEBC=4
Permanent City	City name	Ex. Ahmadabad=0 BARODA(Vadodara)=10 RAJKOT=65 SURAT=68
Board	GSEB,CBSE,ICSE	GSEB=0 CBSE=1 ICSE=2
Performnace	Distinction, First Class, Second Class	Distinction(0)→>7 First Class(1)→between 6 and 7 Second Class(2)→<6
12^{th} Percentage	0<=Percentage<100	

4.5.1 Dataset Features

4.6 Coding Standards

Coding conventions make your code easier to read and debug. Thus, for both our and your benefit, we would like you to use the conventions outlined in this document for your Python code.

Startup

1. Import one thing at a time:

```
import
imaplib
import
stack
from queue import enqueue
```

Spacing

1. Use four spaces instead of tabs. You can set this in Atom by going to Edit> Preferences > TabLength(under the Settingstab). This is particularly important because not only is improperly indented code difficult to read, but python is whitespace sensitive and your

code may not work if you don't follow this guideline.

2. Use two blank lines before any class definition or top-level function
3. Use one blank line between any two methods in a class
4. Put blanks around arithmetic operators, but not before commas, or near brackets or parens:

(3 + 5 * a[4], 33)but not
(3 + 5*a [4] , 33)

Naming

1. Class names start with a capital letter: Stack, Queue, Hashtableor HashTable(the "Cap-Words" convention)
2. Module names are brief and lowercase: stack, graphalgs, . . .
3. Exceptions are classes, so also use CapWords; they end with the word Error ("BadDataError")
4. Functions are lowercase with underscore separators: mst helper
5. Constants: all caps with underscores: MAXOVERFLOW
6. For private data in classes, start with an underscore: my_data

Comments

1. Write a docstring for every class, module, function, and method. Use triple quotes. End with the triple-quote on a line by itself.

```
def gcd(x, y) :
    """gcd: int * int ->int
    Purpose:Computethegcdoftheintegersxandy Example:
    gcd(12, 8) -> 4; gcd(0, -2) -> 2
    """
```

2. The documentation for a function begins with the *signature*, i.e., the name, a colon, the argument types, an arrow, and the return type. That's followed by a description of what the function consumes and produces, a description of the purpose, and illustrative examples, especially cases in which the reader might have doubts about the correct output. The types in the signature may be annotated with brief descriptions:

```
def foo(name, age) :
    """foo: str * int [age in years] -> str [birthday greeting] Purpose: Generate a
    birthday greeting by name and age.
    Example:foo("Fred",21)->"Happy21birthday,Fred" """
```

3. In Python, variables can hold items of an arbitrary type. While this can make for natural coding, it is also makes it easy to give methods unexpected input. For this

reason it is important to be explicit in your method signatures as to what your function produces and consumes. If a method is supposed to receive an object of arbitrary type use any (Note: this is not the same as Object because this does not include things like int and float), or if it consumes a specific type try to use standard names for things (bool rather than Boolean, for instance). Also, if a method does not produce anything, it is designated with a ".".

```
def checkedAdd(name, isCat, myStack) :
    """push: any*bool*stack->.
    Purpose: pushes any and bool onto the input stack in that order Example:
    checkedAdd("Alice", False, myStack) -> the string "Alice" and the bool False are
    pushed onto myStack in that order
    """
```

Runtime Expectations

If we specify a runtime, you must meet that runtime. If we do NOT specify a runtime, you should figure out the most efficient way possible to do the problem. In this case, "most efficient" means big-O. However, you should also consider the constant factor–within reason. This means that you don't need to stress about tweaking your code for the tiniest changes to make it a bit more efficient, especially if it will make your code less clear or concise, but you also shouldn't do obviously inefficient things when there is a much more efficient way that is just as easy to implement as is the less efficient way. For example, if we ask you to write a function that takes as input an integer i and outputs one plus that integer, your function should return i + 1 as opposed to something unnecessarily more complex (and with a bigger constant factor), such as i+sqrt(0+7-7+ 1^2). Both of those would return the correct answer, and both would be O(1), but clearly the second has an unnecessarily large constant factor. Note that we will not necessarily tell you what the best big-O runtime is. Sometimes, it's part of the problem to figure that out yourself.

Testing Expectations

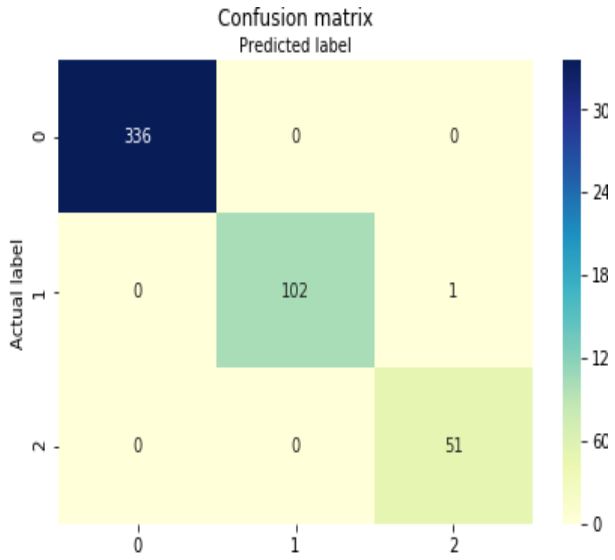
1. If you test your code, you are much more likely to fix bugs in it and hand in 100-percent correct code.
2. If you don't hand in (good) test cases, we may dock points– even if your code works perfectly! What qualifies as a set of "good" test cases, you ask? A good rule of thumb is that test cases should cover both typical and edge-case functionality of your code. This means, among other things, that your test cases should "cover" all of your code (for which we have provided the handy "coverage" module). However, be aware that if you only write test cases according to coverage, you might make a mistake– for example, you may not have accounted for a certain edge case when writing your code, in which case coverage will NOT tell you that you messed up. Also note that there is no "magic number" of test cases that you should hand in. We want your test cases to demonstrate that your code is 100-percent correct, and depending on the complexity of the problem you are asked to solve and/or how much functionality each one of your test cases covers, we can't assign a number to that.

Python's Built-in Utilities

Exercise common sense when using built-in Python methods, data structures, and classes. In general, the rule of thumb says "if using this built-in means that I don't write all of the code that was the point of this problem," then you shouldn't use it. If, on the other hand, using that built-in function does not oversimplify the problem, then feel free to use it. For example, Python has a built-in sorting method that uses an algorithm called "Timsort." If we asked you to implement Timsort in Python, it would NOT be acceptable to use the built-in Python sorting function. Also, keep in mind that it can be temping to use Python built-ins in such a way that the code actually becomes more complicated or slower than it would be otherwise, and try not to fall into this trap. (Yes, you could lose points).

4.7 Result

4.7.1 Random Forest Classification

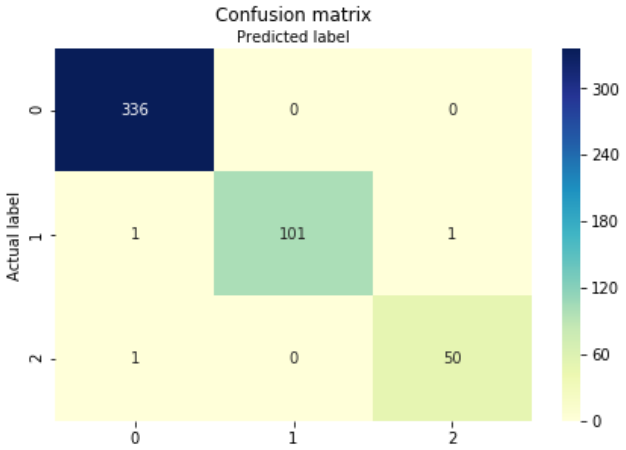


4.7.1.1 Confusion Matrix Of Random Forest

	precision	recall	f1-score	support
0	1.00	1.00	1.00	336
1	1.00	0.99	1.00	103
2	0.98	1.00	0.99	51
micro avg	1.00	1.00	1.00	490
macro avg	0.99	1.00	1.00	490
weighted avg	1.00	1.00	1.00	490

4.7.1.2 Result Of Random Forest

4.7.2 Decision Tree Classification

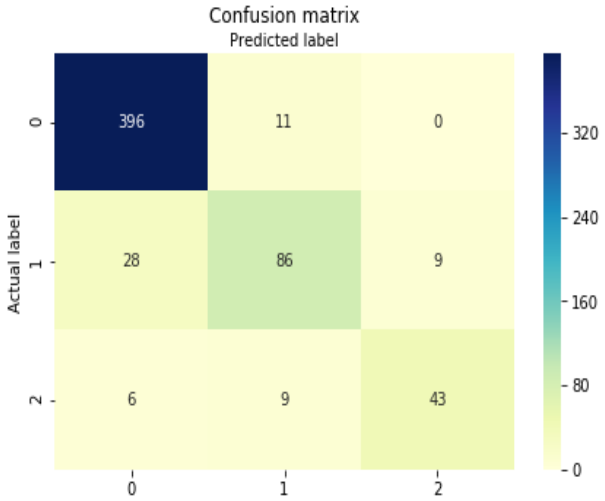


4.7.2.1 Confusion Matrix Of Decision Tree

	precision	recall	f1-score	support
0	0.99	1.00	1.00	336
1	1.00	0.98	0.99	103
2	0.98	0.98	0.98	51
micro avg	0.99	0.99	0.99	490
macro avg	0.99	0.99	0.99	490
weighted avg	0.99	0.99	0.99	490

4.7.2.2 Result Of Decision Tree

4.7.3 KNN



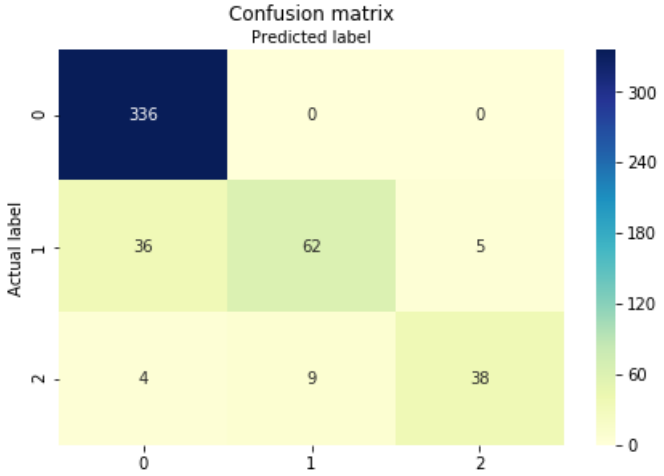
4.7.3.1 Confusion Matrix Of KNN

	precision	recall	f1-score	support
0	0.92	0.97	0.95	407
1	0.81	0.70	0.75	123
2	0.83	0.74	0.78	58
micro avg	0.89	0.89	0.89	588

macro avg	0.85	0.80	0.83	588
weighted avg	0.89	0.89	0.89	588

4.7.3.2 Result Of KNN

4.7.4 SVM(Poly)

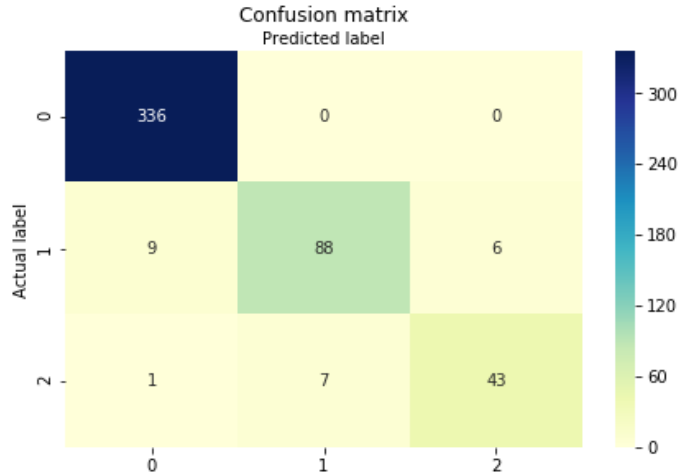


4.7.4.1 Confusion Matrix Of SVM(poly)

	precision	recall	f1-score	support
0	0.89	1.00	0.94	336
1	0.87	0.60	0.71	103
2	0.88	0.75	0.81	51
micro avg	0.89	0.89	0.89	490
macro avg	0.88	0.78	0.82	490
weighted avg	0.89	0.89	0.88	490

4.7.4.2 Result Of SVM(poly)

SVM(RBF)

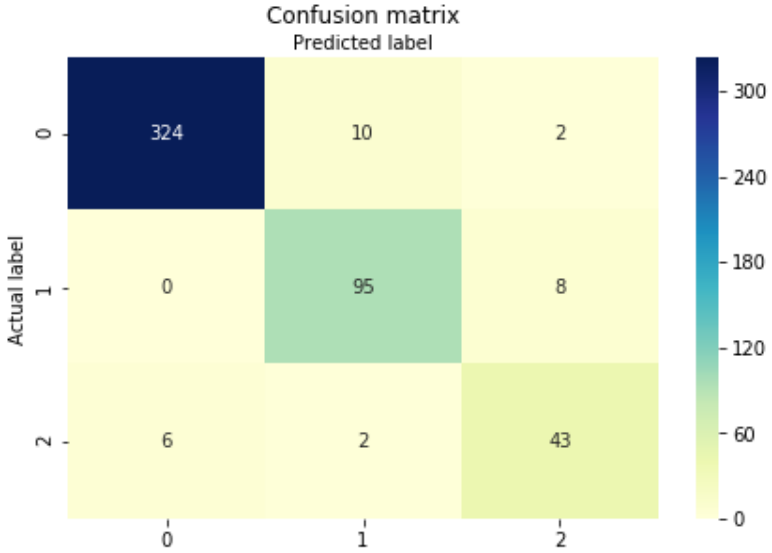


4.7.4.3 Confusion Matrix Of SVM(RBF)

	precision	recall	f1-score	support
0	0.97	1.00	0.99	336
1	0.93	0.85	0.89	103
2	0.88	0.84	0.86	51
micro avg	0.95	0.95	0.95	490
macro avg	0.92	0.90	0.91	490
weighted avg	0.95	0.95	0.95	490

4.7.4.4 Result Of SVM(RBF)

4.7.5 Naïve Bayes

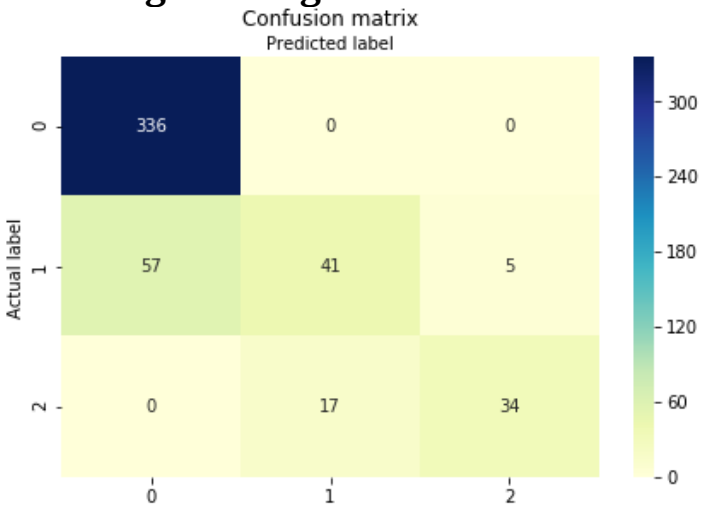


4.7.5.1 Confusion Matrix Of Naïve Bayes

	precision	recall	f1-score	support
0	0.98	0.96	0.97	336
1	0.89	0.92	0.90	103
2	0.81	0.84	0.83	51
micro avg	0.94	0.94	0.94	490
macro avg	0.89	0.91	0.90	490
weighted avg	0.94	0.94	0.94	490

4.7.5.2 Result Of Naïve Bayes

4.7.6 Logistic Regression



4.7.6.1 Confusion Matrix Of Logistic Regression

	precision	recall	f1-score	support
0	0.85	1.00	0.92	490
1	0.71	0.40	0.51	103
2	0.87	0.67	0.76	51
micro avg	0.84	0.84	0.84	490
macro avg	0.81	0.69	0.73	490
weighted avg	0.93	0.84	0.82	490

4.7.6.2 Result Of Logistic Regression

4.7.7 ANN

```
# evaluate the model
scores = model.evaluate(X, Y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

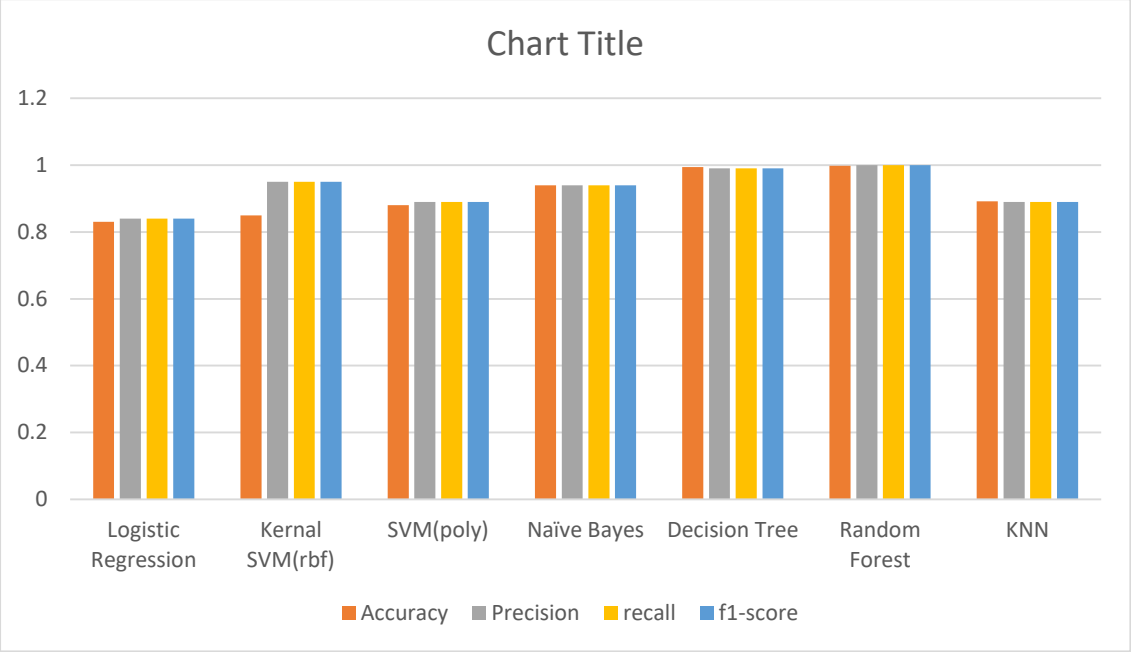
```
1599/1599 [==============================] - 0s 77us/step

acc: 92.27%

Epoch 9/150
1959/1959 [==============================] - 3s 2ms/step - loss: 0.3624 - acc: 0.8
Epoch 10/150
1959/1959 [==============================] - 3s 2ms/step - loss: 0.3499 - acc: 0.8
Epoch 11/150
1959/1959 [==============================] - 3s 2ms/step - loss: 0.3492 - acc: 0.8
Epoch 12/150
```

4.7.7.1 Result Of ANN

4.8 Analysis

	Accuracy	Precision	recall	f1-score	support
Logistic Regression	0.83	0.84	0.84	0.84	490
Kernal SVM(rbf)	0.85	0.95	0.95	0.95	490
SVM(poly)	0.88	0.89	0.89	0.89	490
Naïve Bayes	0.94	0.94	0.94	0.94	490
Decision Tree	0.9938	0.99	0.99	0.99	490
Random Forest	0.9979	1	1	1	490
KNN	0.892	0.89	0.89	0.89	490

4.8.1 Analysis Table



4.8.2 Comparison Of All Classification Algoritjms

Chapter:5 Future Enhancement

Future Enhancement

There are quite a few things that can be polished or be added in the future work.

• We have opted to use machine learning classifies in this project namely the naïve Bayes classification, Logistic Regression, k-Nearest Neighbour, Support Vector Machine, Decision Tree classification, Random Forest classification. There are more classifiers such as the Fuzzy Genetic algorithm and other data mining techniques.

• Though, we have have taken into consideration the academic data of many students, there are still many students and ample amount of input data that could futher be used. With more and more demand for not only student but also performance prediction as a whole, there is alot of data that can be taken into consideration for more accurate results. There is alot of scope for student performance prediction in the data mining world.

Chapter:6 Conclusion

The success of machine learning in predicting student performance relies on the good use of the data and machine learning algorithms. Selecting the right machine learning method for the right problem is necessary to achieve the best results. However, the algorithm alone can not provide the best prediction results. Feature engineering, the process of modifying data for machine learning, is also an important factor in getting the best prediction results.

The aim of this thesis was to compare method selection in terms of their ability to improve the prediction results. Methods used were , naïve Bayes classification, Logistic Regression, k-Nearest Neighbour, Support Vector Machine, Decision Tree classification, Random Forest classification.

As per the analysis we can see that the accuracy of Random Forest Classifier is very higher than all other classification algorithm.

References:

- https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14
- https://www.ijraset.com/fileserve.php?FID=15594
- https://stats.stackexchange.com/questions/306742/how-to-compute-accuracy-for-multi-class-classification-problem-and-how-is-accura
- https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/
- http://notesbyanerd.com/2014/12/17/multi-class-performance-measures/
- https://tampub.uta.fi/bitstream/handle/10024/101646/GRADU-1498472565.pdf?sequence=1
- http://www.cs.kumamoto-u.ac.jp/epslab/ICinPS/Lecture-2.pdf
- https://www.cse.unr.edu/~bebis/MathMethods/NNs/lecture.pdf