Student Performance Predictor

Comparing different Machine Learning Models

Madhusudhan Tungamitta
Computer Science
University of California
Riverside
mtung007@ucr.edu

Balaji Arunachalam
Computer Engineering
University of California
Riverside
barun001@ucr.edu

Shreyas Ghuge
Computer Science
University of California
Riverside
sghug001@ucr.edu

Naman Mittal
Computer Science
University of California
Riverside
nmitt005@ucr.edu

Imran Shahid
Computer Engineering
University of California
Riverside
ishah011@ucr.edu

ABSTRACT

Several ways in the field of data mining tactics are used to create a large volume of electronic data. A new discipline called the Student Performance Analysis is crucial in supporting students and teachers in schools and institutions. The creation of student achievement prediction models to forecast student success in educational institutions is an important area of advancement in Education Data Mining. The purpose and goal of our project was to develop models that can predict a student's performance and grades using and accounting other essential traits like school, sex, age, address, famsize, … , health, absences, etc that impact their execution. We are trying to devise a system that will help students recognize their final grades and improve their academic behavior in order to get a higher grade. Furthermore, using and writing multiple machine learning models we try to compare the performance and output based on a key measure like accuracy. Additionally, we also compare the fulfillment of these algorithms to the one's provided by python's inbuilt Scikit Learn [1] library. We could almost 95% match the accuracy of algorithms from scratch to the ones implemented using the aforementioned Python library.

KEYWORDS

MinMax Scaling, Factorize, Classifier, Random Fores -t, Gaussian Naive Bayes, Decision Trees,Support Vector machines (SVM), Logistic Regression

I. INTRODUCTION

In most educational entities and institutes, predicting student performance has become a pressing need. This is necessary in order to assist at-risk students and ensure their retention, as well as to provide exceptional learning resources and experiences, as well as to improve the university's rating and reputation [6]. Many times, multiple attributes might help us project how a student will evaluate his academic standing. This happens to be crucial, mainly when students diverge from their study plan and thus see a drop in scores and grades. Attributes such as school, age, sex, attendance, health, previous grade record etc are reasonable attributes to understand and predict a student's upcoming grades. Student performance predictor will give the guardian or else the student a report of how the students' final grade sheet would look like. Being a relatively simple project, we'd worked on different Machine Learning algorithms and implemented them from the basic level.

The project aims at discovering a pattern and then providing a prediction of how students will perform based on certain parameters. Precisely, the idea is to assess a student's performance based on estimating the proportion of grades scored from their psychographic

and lifestyle attributes. The only objective is to predict if a student will pass or fail given a vector of attribute values for a student. We have used supervised learning algorithms like Random Forest, Decision Trees, Support Vector Machines, Gaussian Naive Bayes, and Logistic Regression. We worked in pairs to pre-process data by implementing data mining techniques. The data was taken from a kaggle competition. The main reason we chose this dataset is that the data spans from simple data, such as school, sex, age, address, famsize, guardian, traveltime, studytime, failures, absences, passed etc. which are absolutely relevant to the project and can give us the required information to train our models. We could hence explore different machine learning procedures and learnt to appropriately apply different data mining techniques. The data comes in a CSV file format. The data in each row consists of values of different types, including numbers and text. No additional test data is given so we divided this data for testing and validation purposes.

II. RELATED WORK

There have been multiple instances of thorough research conducted on this topic. Many of them used different methods to solve the titled problem statement and hence computed relevant scores to rate the working. However, the most number of techniques used in a single implementation was 3. In order to solve this problem, authors J. Dhilipan et. al proposed solutions [3] and conducted studies using algorithms such as Binomial logical regression, Decision tree, Entropy and KNN classifier. They built a confusion matrix and presented Accuracy scores to showcase their implementation's performance. The report also gave us the idea of measures such as precision, recall, f1 scores support for each procedure.

Paper presenters [2] along with Sitha Ram considered analyzing the performance of the students and therefore gave extra attention to certain low performing ones. Primarily they focus on using Logistic Regression and Random forest algorithms to construct their system. This inspired us to look more in depth in the Random Forest algorithm as it performed well and delivered a promising accuracy.

According to a paper [4] that focused on EDM i.e Educational Data Mining, the authors considered midterm grades, Department Data and Faculty Data. of students as data to predict their future performance. To end with, this study aids in the early diagnosis of students who are at risk of failing, as well as the identification of the most successful machine learning methodologies.The results show that the proposed models could formulate a classification accuracy of around 75%.

Authors of this paper [5] worked with functional near-infrared spectroscopy to determine students' performance. The PFC's fNIRS data was utilized to create predictions for a student's reactions in this study. Machine learning techniques were used to obtain these predictions from the data. They further employed random forests and penalized logistic regression techniques in particular to balance out working of those techniques. These algorithms allow for the decoding of present data structures as well as the formulation of prediction rules for fresh observations.

Following the papers from the above related work already performed in the domain, we have chosen algorithms mentioned in the prior content, to (1) get a good grasp of what we could explore under the field of Data mining and Machine Learning Algorithms. (2) have an accuracy score of algorithms implemented from scratch to be as close as the one’s enacted using the inbuilt Scikit Learn [1] Library of Python. (3) and predict and classify students based on some of the attributes on which the models were trained on.

III. DATASET OVERVIEW / PREPROCESSING

The information was gathered through a kaggle competition. The main reason we chose this dataset is that it contains basic information such as school, gender, age, address, family size, guardian, traveltime, studytime, failures, absences, and passing grades, all of which are extremely relevant to the project and can provide us with the information we need to train our models. This allowed us to investigate various aspects of machine learning and use various data mining approaches properly. A CSV file is given and according to each row's data the tuples were made up of several sorts of values, such as integers and text. Because no additional test data is provided, we used

the provided data and split the same for the purposes of testing and validation.

Preprocessing data in order for them to work with machine learning techniques is an essential step. ML models are often developed and evaluated on a massive dataset. However, we had a Kaggle dataset in csv format that we could use. We scaled down and normalized the values in numerous characteristics with a variety of records. We utilized the MinMaxScaler provided by Scikit Learn to do this. We also used the Factorize technique from pandas to clean the dataset that contained textual data. We performed the following data pre-processing tasks so as evaluate our models:

- *MinMax Scaling* - By scaling each feature to a certain range, MinMax scaling alters properties. "This estimator scales and translates each feature separately such that it falls inside the provided range on the training set, such as zero to one" [1. Transforming the data into a certain range made it easier to work with the algorithms and, as a result, to make greater use of the results.

- *Factorize* - Multiple classes and property values that are not number encoded are common. For example, textual data, acronyms, one-word binary classes, and so on. This information must be converted to a numeric value before being fed into the machine learning models. "Encode the item as a categorical variable or an enumerated type. When all that counts is distinguishing different values, this approach is handy for getting a numeric representation of an array." [7].

- *Basic data cleaning* - removing not applicable values in addition with blank and missing values.

IV. PROPOSED METHODS

We tried and implemented 5 Machine Learning algorithms to imitate the working of the one's that Scikit Learn provides along with the goal of successfully predicting if a student will pass or fail given certain academic parameters. In our group, every member took interest in learning about 2 algorithms from the 5 selected one's and therefore everyone worked in pairs to implement an algorithm. This not only gave a chance to explore more than 1 machine learning technique but also gave a far better comparison of how parameters are used and tweaked between them.

IV. I Random Forest Classifier

i. Overview

A random forest classifier is made up of multiple interconnected decision trees. Each tree makes a forecast, and the class with the most votes is our model's guess. The max samples option defines the number of sub-samples if bootstrap=True (the default); otherwise, the whole dataset is utilized to create each tree [8].

1: **procedure** RANDOM FOREST
2: **for** 1 to T **do**
3: Draw n points D_l with replacement from D
4: Build full decison/regression tree on D_l
BUT: each split only consider k features, picked uniformly at random
new featues for every split
5: Prune tree to minimize out-of-bag error
6: **end for**
7: **Average** all T trees
8: **end procedure**

Fig 1: Random forest algorithm [1]

The reason for picking random forest is that it is a particularly effective model since a large number of associated trees working together will outperform any of the individual models. To create the model, we use the Random Forest Classifier from sklearn. Ensemble with the parameters n_estimators, max_depth, and random state. Random forest algorithm was the best possible machine learning algorithm we could devise.

ii. Comparing implementation with Scikit Learn

We experimented with several values for n_estimator and max depth. With the above-mentioned settings, we can get the most out of the RandomForest Classifier in terms of accuracy. More is described in the Implementation Correctness Report. While we used the inbuilt library where it achieved an accuracy of 65 %, we could come as close as getting the test accuracy as 63.29 % by implementing the Random Forest Procedure from scratch. Although there was a dip in the accuracy scores when comparing, we had chances to explore multiple different hyper-parameters during the process.

[1] Image source https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote17.html

IV. II Gaussian Naive Bayes Classifier

i. Overview

Naive Bayes on Bayes' theorem, a Bayes classifier is a sort of "probabilistic classifier." The word nave comes from the fact that this supervised machine learning technique is predicated on high feature independence. Naive Bayes classifiers require a linear number of parameters in proportion to the number of variables (features/predictors) in a learning task. Unlike many other forms of classifiers, maximum-likelihood training may be accomplished by evaluating a closed-form expression in linear time rather than through time-consuming iterative approximation. The likelihood of an event is described by Bayes' theorem, which is based on previous knowledge of conditions that may be essential to the event.

1. for q = 1…w // loop for each mining models element
2. μ[q] = 0; // initialization of mining models elements
3. end for;
4. for j = 1…m // loop for each row
5. μ[d[j,p]]++; // increment number of row for value $x_{j,p}$ of object x_j;
6. for k = 1 …p-1 // loop for each column
7. $\mu[\varphi(k-1)+(d[j, k]-1)\cdot\varphi(0)+ d[j, p]]$++; // increment number of rows with value $x_{j,k}$ // and value $x_{j,p}$, where $\varphi(k)=s+\sum_{q=1}^{k}(|T_q|\cdot s)$
8. end for;
9. end for;

Fig 2: Naive bayes basic algorithm [2]

For the below formula, σ = standard Deviation, μ = Mean

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Fig 2.1: Naive bayes basic algorithm [3]

Using a Gaussian NB was easier to implement and resulted in a higher accuracy score. With each step, we also computed the probabilities and updated the previous values. The Bayes theorem, which employs given probability P(A | B), is the foundation of Gaussian Naive Bayes. As a result, we have enough characteristics to distinguish the classes, lowering the entropy of the decision and allowing us to make very precise and accurate predictions.

ii. Comparing implementation with Scikit Learn

We summed up the working of Naive Bayes algorithm at training accuracy score of 73.73 % and for testing accuracy of 70.88 % and when we tried to implement naive bayes algorithm from scratch we got an accuracy of 68.35 % and precision score of 73.01 % with libraries and 65.62% when we implement from the scratch.

IV. III Decision Tree Classifier

i. Overview

Ross Quinlan created the C4.5 method for creating decision trees. It's an improvement on the ID3 algorithm. For both classification and regression, a decision tree is employed. It's a tree structure that looks like a flowchart, with each internal node representing a test on an attribute, each branch representing a test result, and each leaf node holding a class label. Gain Ratio is used as an attribute selection measure in this ML algorithm. Expected information needed to classify a tuple in dataset D is

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Information needed after using attribute A to split D into v partitions

$$Info_A(D) = \sum_{j=1}^{v} |D_j| / |D| \times Info(D_j)$$

Potential information generated after using attribute A to split D into v partitions:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} |D_j| / |D| \times \log_2 |D_j| / |D|$$

Moreover, the Gain Ratio for attribute A is given as below:

$$GainRatio(A) = Gain(A) / SplitInfo_A(D)$$

[2] Image source https://www.researchgate.net/figure/The-Naive-Bayes-algorithm-pseudocode_fig3_336219050

[3] Image source https://iq.opengenus.org/gaussian-naive-bayes/

We used this classifier in our project since the main focus of the problem statement is binary classification, and we have a 50/50 possibility of students passing or failing. The DecisionTreeClassifier from the sklearn python package is used. We experimented with a variety of parameters and worked with a variety of values. We modified the tree's values, specifying the max_depth, min_samples for leaves, and min_sample for split parameters, and predicted the output using the model. The decision tree made sense since the default classification rate for our problem is 50%, and the decision tree will, by definition, offer us a larger and more accurate result.

ii. Comparing implementation with Scikit Learn

We observed a much greater gap in the accuracies from the one we casted during the midterm report. While we worked on Decision Tree using the library, we could get a testing accuracy of 80 % and testing accuracy of around 72 %. After we tweaked parameters in the algorithm we ultimately were able to reach an accuracy of 95 % while training and 62 % when we tested the samples.

IV. IV Support Vector Machine Classifier

i. Overview

SVM is a widely used discriminative classifier for both linear and non-linear dataset classification. SVM determines the best hyperplane for separating one class from another. It employs a nonlinear mapping to transfer nonlinear data to the appropriate higher dimension, and then finds the best hyperplane. Support vectors, which are data tuples that impact the decision boundary, are used by SVM to locate the hyperplane. As a result, the largest margin (distance between two classes support vectors) is determined using SVM.

We attempted to fit the data using SVM and returned the best fit line that categorizes the data. To divide the two sorts of data points, a variety of hyperplanes can be employed. Our aim was to find the plane with the largest margin, or distance between data points from both classes. Maximizing the margin distance provided some reinforcement, making it simpler to categorize subsequent data points [9]. We took the liberty of using SVM's ability to distinguish classes in a more efficient and accurate manner by leveraging the hyperplane's nearest points.

Algorithm 1 Pseudo-code of SVM algorithm

Inputs:Determine the various training and test data.
Outputs:Determine the calculated accuracy.
Select the optimal value of cost and gamma for SVM.
while (stopping condition is not met) **do**
    Implement SVM train step for each data point.
    Implement SVM classify for testing data points.
end while
Return *accuracy*

Fig 3: SVM algorithm [4]

The most basic equation of SVM was used so as to fine tune the values and get the correct result.

$$y \leftarrow sign\left[\sum_i \alpha_i y_i K(x_i, x) + b\right]$$

Furthermore, we made use of the fact that the hyperplanes assisted us in classifying and segregating the classes independently. We decided to use the hyperplane to demonstrate that the data is separable since features may have semantic commonalities and hence be grouped based on only a few characteristics out of many, eventually resulting in two superclusters with the classification labels Pass or Fail. As a result, an SVM, which is already an outstanding tool for binary cluster classification, was well suited to predict the grades (pass or fail) of students.

ii. Comparing implementation with Scikit Learn

We tested with various parameter settings. While we were able to reach a train accuracy of 92 % using the Scikit learn library, we were able to get as close as 60.72 % testing accuracy by creating the Support Vector Machines from the beginning. Additionally, we had a testing accuracy of 72 % employing the python library.

IV. V Logistic Regression

i. Overview

Logistic regression is a supervised classification approach for predicting a categorical dependent

[4] Image source https://www.nature.com/articles/s41598-020-71502-z

variable's likelihood. The dependent variable is a binary variable that holds data coded as true or false, with 1 indicating true and 0 indicating false. When the dependent variable denotes the classification of the target, Logistic Regression is one of the most useful machine learning methods. forecast the likelihood of outcomes, which in our case implies whether the student will pass or fail. depending on the data that has been pre-processed We are one of the many forms of Logistic Regression. Multinomial Logistic Regression, which allows for two or more categories of dependent or independent variables. without any sorting of result variables There are no assumptions about linearity, normalcy, or homoscedasticity. As a result, it is one of the most appealing algorithms.

1. For i←1 to k
2. For each training data instance d_i:
3. Set the target value for the regression to
   $z_i \leftarrow \frac{y_j - P(1 \mid d_j)}{[P(1 \mid d_j) . (1 - P(1 \mid d_j))]}$
4. nitialize the weight of instance d_j to P (1 | d_j). (1 – P (1 | d_j)
5. finalize a f(j) to the data with class value (z_j) & weights (wj)
   Classification Label Decision
6. Assign (class label:1) if P (1|d_j) > 0.5, otherwise (class label: 2)

Fig 4: Logistic Regression algorithm [5]

The random state ensures that the splits are consistent with the data we generated. If the user does not define random permutation, Scikit-Learn will use this as a seed for a random number generator. To estimate the changes in the second derivative matrix with the gradient evaluations, the solver "lbfgs" (Limited memory Broyden Fletcher Goldfarb Shanno) is employed. It's used to save memory by just storing the most recent changes. This will be the default solution for Multinomial Logistic Regression in Scikit-Learn. The fit() function describes the relationship between the collection of predictors and the continuous answer using the ordinary least squares approach.

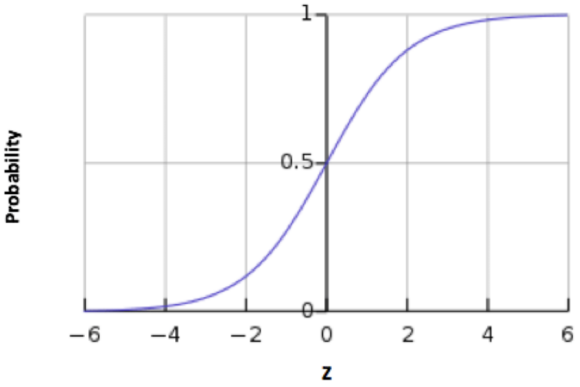

Fig 4.1: Graph of Sigmoid function [6]

ii. Comparing implementation with Scikit Learn

We could see the average accuracy scores for Logistic Regression as well. It performed almost similarly to other designs we followed. Comparing our output to Scikit learn's algorithm, we had a deficit of 6 %. During the usage of the out-of-the-box version of the algorithm we were able to produce training accuracy of 73 % and testing accuracy close to 70 %. We managed to replicate the algorithm and availed a test accuracy of 64.5 %.

V. EXPERIMENTAL EVALUATION

We used 4 metrics namely accuracy, precision, recall, and F1 score for comparing the classifiers. These measures were helpful to know which algorithms were impactful in the classification and how they performed. Below is a quick summary of all the metrics we used:

- *Accuracy*: It is the ratio of correctly predicted observations to the total observations.

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

- *Precision*: It is the ratio of correctly predicted positive observations to total predicted positive observations.

$$Precision = \frac{TP}{(TP + FP)}$$

[5] Image source https://www.researchgate.net/figure/Pseudocode-of-logistic-regression_fig15_343120305

[6] Image source https://www.researchgate.net/figure/Standard-logistic-sigmoid-function_fig11_336216799

- *Recall*: It is the ratio of correctly predicted positive observations to all positive observations in the actual class.

$$Recall = \frac{TP}{(TP + FN)}$$

- *F1 Score*: It is the harmonic mean between precision and recall.

$$F1 = 2 \times \frac{(Recall \times Precision)}{(Recall + Precision)}$$

1. Decision Tree Algorithm -
    a. Accuracy: 62 %
    b. Precision: 0.71
    c. Recall: 0.76
    d. F1 score: 0.73
2. Random Forest Algorithm
    a. Accuracy: 63.29 %
    b. Precision: 0.72
    c. Recall: 0.76
    d. F1 score: 0.74
3. SVM Algorithm
    a. Accuracy: 60.76 %
    b. Precision: 0.65
    c. Recall: 0.98
    d. F1 score: 0.78
4. Naive Bayes Algorithm
    a. Accuracy: 68.35 %
    b. Precision: 0.77
    c. Recall: 0.89
    d. F1 score: 0.76
5. Logistic Regression
    a. Accuracy: 64.5 %
    b. Precision: 0.70
    c. Recall: 0.83
    d. F1 score: 0.76

VI. CONCLUSION

To recap, we pre-processed the dataset so that we could work effectively with several machine learning methods. After cleaning the data, we used Python's sklearn package to experiment with several machine learning techniques. We were able to experiment with a variety of parameters and fine-tune their values to get the best accuracy for each of the individual algorithms. The following table compares the accuracies of the various methods listed above. All of the machine learning models had comparable findings, as we expected. On the test samples, the accuracies ranged from 0.6 to 0.7.

REFERENCES

1. https://scikit-learn.org/stable/ [last accessed: 5/27/22]
2. M. S. Ram, V. Srija, V. Bhargav, A. Madhavi and G. S. Kumar, "Machine Learning Based Student Academic Performance Prediction," 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), 2021, pp. 683-688, doi: 10.1109/ICIRCA51532.2021.9544538.
3. J. Dhilipan *et al* 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* 1055 012122
4. Yağcı, M. Educational data mining: prediction of students' academic performance using machine learning algorithms. Smart Learn. Environ. 9, 11 (2022).
5. https://www.frontiersin.org/articles/10.3389/fnhum.2021.622224/full [last accessed: 5/26/2022]
6. Prediction of Students performance [online], Article number 27, 2019. https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-019-016-0-3
7. https://pandas.pydata.org/docs/reference/api/pandas.factorize.html [last accessed: 04/29/2022]
8. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [last accessed: 04/30/2022]
9. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [last accessed: 05/02/2022]