Interpretable MOOC dropout prediction using different Ensemble Methods and XAI

by

Labib Hasan Khan
19101014
Mohammed Ashfaqul Haque
19301002
Esaba Ahnaf Ibrahim
20301422

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
School of Data and Sciences
January 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ashfaq

Mohammed Ashfaqul Haque

19301002

ESABA AHNAF IBRAHIM

Esaba Ahnaf Ibrahim

20301422

Labib

Labib Hasan Khan

19101014

Approval

The thesis/project titled "Analysis of MOOC dropout using Deep Learning and Machine Learning models." submitted by

1. Labib Hasan Khan (19101014)
2. Mohammed Ashfaqul Haque (19301002)
3. Esaba Ahnaf Ibrahim (20301422)

Of Fall, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 17, 2023.

Examining Committee:

Supervisor:
(Member)

Moin Mostakim

Moin Mostakim

Senior Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

[illegible]

Dr. Muhammad Iqbal Hossain

Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Associate Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Massive open online course (MOOC) has been around for a while, but started to gain traction since 2012 when Coursera was established. MOOCs use pre-recorded lectures and scheduled weekly tests to provide content and access to students over the internet. Even though there was a high expectation that it would revolutionize the education system, due to the mode of one-way content delivery, the goal was too far-fetched. The flexibility in deadlines and no restrictions of classroom exams meant students were not bound to finish on time. Hence, most students did not finish the course and dropped out. The dataset used in our research was the KDDCUP 2015 dataset, which was publicly available by the organizers of XuetangX platform. We used about 12 features namely browser access, navigate, average chapter delays, server sequential etc to comprehend the possibility of dropout. In this paper, we aim to predict dropout of a learner so that it can be prevented through manual interaction. Additionally, we have implemented XAI to interpret our models to suggest MOOC platforms which feature impact dropout the most. We used different ensemble learning techniques, namely voting classifier and stacking. Our voting classifier uses five of our best performing machine learning models. Then we evaluate the model by using multiple metrics such as precision, recall, F1-score, ROC curve and AUC score. Finally, we managed to obtain a recall of 97.636% with stacking and f1-score of 91.603% with hard voting classifier.

Keywords: Machine Learning; Deep Learning; MOOC; Dropout Prediction; Ensemble Learning; Explainable Artificial Intelligence (XAI);

Acknowledgement

Firstly, all praise to the Great Allah, for whom our Final Thesis have been completed without any major interruption.

Secondly, to our supervisor Moin Mostakim sir and co-supervisor Muhammad Iqbal sir for their kind support and advice in our work. They helped us whenever we needed help.

And finally to our parents without their throughout support it may not be possible. With their kind support and prayer, we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ABC	AdaBoost Classifier
AME	Attention Meta Embedding
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DTC	Decision Tree Classifier
GBM	Gradient Boosting Method
GNB	Gaussian Naive Bayes
GRU	Gated Recurrent Unit
KNN	K-Nearest Neighbour
LIME	Local Interpretable Model-Agnostic Explanation
LR	Logistic Regression
LSVC	Linear Support Vector Classifier
LTSM	Long Short Term Memory
MLP	Multilayer Perceptron
MLR	MultiNomial Logistic Regression
MOOC	Massive Open Online Courses
NLP	Natural Language Processing
ONet	Object Network
RFC	Random Forest Classifier
SHAP	Shapley Additive Explanation
SMOTE	Synthetic Minority Over-Sampling Technique

SVM	Support Vector Machines
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Motivation

Now more than ever, education has become crucial in aiding better living standards and reducing poverty in most developing countries. With the rapid development of internet technology, there has been a rise of online learning sites with the aim of delivering high quality education to the masses. To cope with the ever-changing job requirements, such e-learning platforms have been developed to assist them in bridging the gap between the talents that the industry requires and the skills that they acquire at colleges and universities. MOOC was first coined in 2008 by Dave Cormier and Bryan Alexander, which stands for Massive Open Online Courses. Due to its online and open nature, it has become exponentially popular worldwide, as anyone is allowed to participate in it and access state-of-the-art courses from world leading universities.

Compared to traditional classrooms, MOOC is quite different. Firstly, it overcomes any environmental constraints in space and time as learners can not only choose where to study but also learn at any time according to their convenience. Each student can progress at their own pace and can access all the resources all the time, which previously could only be available during school hours. Contents such as video recordings can be played repeatedly and with the help of live forum discussion they can interact with fellow students and teachers to clear any confusion. MOOC is based on a connectivist approach, so any knowledge is distributed over the network to all participants. Students can study in their chosen field without restriction, primarily because MOOC is free and have flexible deadlines.[22]

Since its inception, MOOCs have evolved from simple online platforms to complex learning management platforms. Paving the way for a new era of education, several organizations have taken their concept and established their own platforms. Later, in early 2012, professors from Stanford University founded Coursera, an independent for-profit company. Following their footsteps, other similar initiatives such as Udacity, Udemy, Edx, Iversity, MiriadaX have been launched since then across the US and Europe. With 58 million participants globally, divided across more than 7,000 courses, and more than 700 partner universities, a 2016 report from Class Central demonstrated just how much MOOCs have contributed.[10]

1.2 Problem Statement

Although MOOCs have greatly improved high quality education access to the masses, there is a growing concern for their further development as a high number of students are dropping out without completing the courses. It is becoming increasingly difficult for instructors to monitor all the students at risk at the same time and prevent them from withdrawing. This is where machine learning comes into play, helping us to predict dropout of students based on their interaction data with the course materials and resources, and thus intervene them appropriately before they dropout.

According to [25], learners' household condition or environment, earnings, and employment state (work schedule) are some of the reasons that presumably influence the dropout of online courses. From research paper [13], it is observed that students experiencing little to no empathy from their family members or confronting difficult circumstances at work places are more prone to abandon e-learning in contrast to an individual undergoing right guidance from their respective families. Hence, all these factors play a critical impact on the dropout rate from MOOC. [18][19][8]

The research paper [11], however, expressed that e-learning finishing off numbers may have a relation to the course duration, hence, long-lasting critical courses are seen to have higher incompletion rates. Subsequently, the approaches adapted for teaching a lesson, critical thinking, devotion towards the online lesson and duration of the course work are redeemed as four criteria that have been discovered and classified under the 'course' category. Furthermore, lack of enthusiasm among students might be explained by the fact that the course is free, thus students are not obligated to complete the online course resulting in an increased number of dropouts in MOOC.[25]

The recent pandemic also demonstrated the necessity of MOOCs as education was fully moved to online. A lot of schools and universities were not prepared to tackle such difficulties, which further reinforces the idea of crucial development of MOOCs to the point that every level of education can be serviced. Some courses of BRAC university also require taking part in MOOCs which shows the significance of its contribution to a student's academic journey.

Additionally, almost all previous research works related to MOOC dropout did not try to explain their model's decision and find the most important factors that contributes to dropout. We have implemented XAI to explain our models and suggest improvements based on the most significant features.

According to [20] the following points show some factors affecting dropout rate:

Student related:

- Lack of motivation: Future economic gain, personal and professional identity development, challenge and achievement, enjoyment and fun are all variables that impact their motivation. The survey found that reasons for enrolling in MOOC ranges from enjoyment, general interest in a topic to deciding whether to pursue higher education or simply because they cannot afford formal education.

- Lack of time: Another aspect that has a significant impact on course completion is time management. Watching all the lectures and completing all the quizzes and assignments appears to take up far too much of a student's time.
- Insufficient background knowledge and skills: Inadequate background knowledge and lack of skills, especially mathematical expertise, are common cause for not being able to finish a course. Reading, writing, and typing abilities are essential in completing a course.

MOOC related:

- Course design: Course design generally consists of three parts, course content, structure and delivery method. Among them, students complained about the complexity and the technicality of the course contents, preventing them from completing it. In addition, the language used were also found to be intricate and having too many modules did not help either.
- Isolation and lack of interactivity: Surveys showed that lack of interactivity of students in discussion with their lecturers resulted in getting poor feedback. Furthermore, non-existing communication between students gave them a feeling of isolation.
- Hidden costs: Even though it is free to access the courses, there are hidden costs involved such as paying for certificates and textbooks suggested by the instructors, which often leads to withdrawal from the course.

Challenges of Dropout Prediction Using Machine Learning [20] :

- Managing big masses of unstructured data: Data is unstructured and in huge amounts. As a result, during preprocessing, the data needs to be manually cleaned. We can not work with such missing data on HMM (Hidden Markov Model).
- Data variance and imbalance: Data variance leads to imbalance classes. The number of dropouts is significantly higher compared to non-dropouts. So when the model is trained on it, the prediction is biased in favour of dropout so if tested, non-dropout students can be labelled as drop-out.
- Availability of publicly accessible dataset: The two main publicly available dataset is the KDDCUP 2015 and the OULAD dataset. The dataset used by other researchers was either requested by the particular scholar or taken through polling.
- Inconsistent data structure: Dataset structure is inconsistent among MOOC platforms. Models trained on one set of dataset can not be used on another platform's dataset because of these inconsistencies.

- Student schedule related challenges: Satisfying the different preferences of student's timetable is yet another challenge that needs to be addressed.

Now that we have identified a need to predict dropouts, it is also important to figure out why that prediction is made so that we can deal with the root problem directly. If we can find out which features are responsible for students dropping out, it would give us a much better understanding of the underlying issue and help us to focus on them. However, the challenge lies in the fact that machine learning and deep learning models are a kind of black box; meaning they cannot be understood simply by looking at their parameters. This is where we need interpretability to comprehend the decisions made by these models. In the paper [17], Miller defined interpretability as the extent to which a human can understand the cause of a decision. In our case of MOOC dropout problem, it is not enough to know the prediction only, but also to know the 'why' behind that prediction. Recently, explainable artificial intelligence (XAI) has established its place in AI, with advancements in explainability and applications in areas like insurance, medicine, and others. To be relevant, the interpretation of the system must yield information about the model's mechanisms and predictions, a representation of the model's discrimination rules, or clues about what can disturb the model [21]. There is a large degree of variation between interpretability and explainability. As mentioned before, interpretability refers to how well an attribute that the model possesses passively makes sense to a human observer. This quality is often referred to as openness. In contrast, explainability can be thought of as an active feature of a model, referring to everything the model does or does not do to explain or detail its underlying workings.[23]

1.3 Objective and Contributions

This research aims to develop an architecture that can find the core reasons behind student dropout and accurately predict MOOC dropout while also being relatively fast.

1. To deeply understand the dropout problem of MOOCs.
2. To develop a model for MOOC dropout prediction
3. To evaluate the model.
4. To interpret our model's decision-making to figure out which features are having the most impact.
5. To make suggestions for how to improve MOOC platforms.

Chapter 2

Background

2.1 Literature Review

MOOCs have modernized education by allowing anyone who chooses to learn and build vital skills for their future for free. It is a platform where instructors can distribute their knowledge to thousands of people. However, the alarming challenges of preventing mass dropout of students still remain to be addressed. The ultimate goal is to be able to learn and anticipate early withdrawal behaviour with high accuracy while being computationally less taxing.

2.1.1 Defining Dropout for MOOCs

A proper definition of dropout is required in order to classify them using various methods from machine learning and deep learning. Dropout is defined differently by different scholars; some define it as failing to complete a course or failing to receive certification. Others have based it on the students' learning behaviour, such as whether they have watched the lectures long enough or are still stuck on previous weeks. Authors have also used clickstream on forum posts to determine dropout of learners.[24][12]

2.1.2 Machine learning based dropout Architectures

For most MOOC dropout prediction, machine learning techniques have been adopted, while deep learning models are becoming more popular as they provide more insights to how students are headed towards that path. CNN, SVM, decision trees, logistic regression models are seen widely in dropout research. However, deep learning techniques are essential to learn how attention impacts dropout behaviour, as models like ONet and CLSA have revealed.

2.1.3 Related Works

This section tries to critically evaluate previous relevant works on MOOC dropout. We look at the many strategies utilized to get the prediction results and show where there is room for improvement, such as accuracy and speed.

According to the researchers [35], a machine learning algorithm could be implemented to inform learners to complete course work they were supposed to completed in an estimated timeframe. The number of days to be evaluated may be reckoned by comparing accuracy to days post-registration, in which the researchers will analyse how productive or how many days it plans to operate based on the data received after applying the algorithms. Future studies could look at more recent data and include the pandemic scenario's impact on student behaviour. In addition to that, future research could also include more data to reflect the diverse effect that Covid-19 had on the students. Python programming has been implemented and AdaBoostClassifier (ABC), RandomForestClassifier (RFC), DecisionTreeClassifier (DTC), LinearSVC(LSVC), GaussianNB (GNB) and LogisticRegression (LR) models were used.

In light of the paper's research [15], the study included a dropout prediction system for MOOC platforms. The strategy aims to make use of data from MOOC platform learning activities. They employed a two-layer cascade classifier with three machine learning classifiers for dropout prediction: Random Forest (RF), Support Vector Machine (SVM), and MultiNomial Logistic Regression (MLR). According to experimental data, the proposed strategy improves performance in terms of precision, F1 score, AUC, and accuracy. Furthermore, the proposed technique has a lower recall rate than discrete methods, which is acceptable. The authors will continue to develop and improve the current strategy in future studies in order to achieve a higher efficiency without diminishing recall.

In this paper [36], the model that the authors chose was an Artificial Neural Network(ANN) with nine input nodes for nine features and one node as output. ANN is used as we do not know the relation between inputs and output. The output was normalized to give 0 or 1 as output. After testing many combinations to see which gave the best results, they opted for two hidden layers with 8 neurons on each layer. The author also implemented the same model with sentimental analysis on forum posts, which was not done previously. The ANN showed a higher AUC score of 0.93 to 0.98 over several weeks while other models such as KNN, SVM, Decision Tree all had lower scores of 0.7 to 0.9. ANN with sentimental analysis always outperformed the simple ANN.

According to this paper [29], the authors presented an attention meta embedding based deep temporal network to predict dropout in MOOCs based on the learning pattern of each student. To interpret the learners' engagement, they proposed a new deep attention learning network (ONet). It consists of two stages: the first stage develops an Attention Meta Embedding (AME)-based dense representation for each time step at the day level precision. The temporal categorization of the extracted meta embeddings, which are used as features, is the second phase of ONet. Because of its faster and superior performance, the authors chose bidirectional GRU over LSTM to circumvent the latter's restricted time steps problem. In comparison to baseline findings, their model outperformed other models on the KDD cup 2015 and XuetangX datasets.

In the research work [31], the FWTS-CNN, a modified convolutional neural network model integrated feature weighting and behaviour time series, which played an essential role in dropout prediction. The first phase required selecting features and building decision trees with 7 learning behaviours. After ranking these features on their importance, they are weighted on the basis of the raw data. Subsequently, a two-dimensional time series matrix is formed on the weighted dataset based on the features of the learning behaviour. After going through the CNN layer, the logistic classifier is utilized to get the classification results that correspond to the student behavioural characteristics. This model proved to be superior and outperformed several other traditional machine learning models.

The authors in the research paper [33] proposed a novel model called CLSA which practically combines CNN, LSTM networks and attention mechanisms, which was their primary focus for a better prediction. To generate a prediction, their model pulls local features from the original data, combines them with time series data, and then multiplies them with feature-wise weights. The time series matrix generated by the CNN layer is used by the LSTM layer to assess the temporal relationship between students' weekly behaviour characteristics and apply weights based on the importance of the behaviour utilizing the attention mechanism. The static attention mechanism generates the weights of each feature information and multiplies them with the input feature information for adaptive learning. Because the data can be represented in a vector format, the authors employed static attention to assign greater weights to more essential information, improving the model's efficiency.

In the following paper [30], the knowledge obtained in the field of natural language processing(NLP) is applied in MOOC dropout prediction and the authors tried to show the effectiveness of using attention as an alternative to LSTM and adding conditional random field in a neural network architecture to solve the problem. To increase its ability of representation, a feature vector is produced using statistical features, accompanied by a linear transformation layer and a self-attention layer. The linear transformation layer helps to map statistical features into higher dimension space, which is found to give better performance as demonstrated by NLP field. Avoiding data leakage from the future, a separate attention strategy is applied, followed by a masked self-attention layer. A probability model called Conditional Random Field (CRF) is used to make better use of the information and predictions from the past. Both of these layers overcome the encoder-decoder drawbacks and provide a more accurate representation of the features. The output goes through the CNN layer to generate time-unit level representation. An additional self-attention layer prevents interference of future information after revealing the relationship between the time series. The output is then converted to a label vector using transformation, normalization, and SoftMax layers while avoiding the gradient-vanishing problem. Going through another CRF layer produces the final output, which proves the superior performance of the model compared to other conventional models.

Interpretability can be classified into several categories; one such criterion is whether it is intrinsic or post hoc. Intrinsic refers to machine learning models that are interpretable due to their simplicity in their structure, while post hoc refers to interpretation after training of the model. The drawback with intrinsic interpretable models is that they have poorer performance due to being restricted to only one type of model. Decision tree is an example of intrinsic interpretable model. The flexibility of model-agnostic interpretation methods is a significant benefit when compared to the specificity of model-specific interpretation methods. As a result, machine learning developers do not need to worry about using a specific model, instead, they are free to pick their preferred model as the interpretation can be done on any type of model. It also provides the added benefit of using any explanation form, whether it is a linear formula or graphical representation [38]. The scope of interpretability can be divided into parts, local and global. In local interpretation, the prediction might merely depend on the linearity and repetitiveness of features, rather than having a complicated relationship with them. It predicts a single instance for the given input. On the other hand, in global interpretation, the level of interpretability required to comprehend the reasoning behind the model's choices by taking into account all the features and learned parameters and weights needs the trained model, the algorithm, and the data. The capacity to assess a model on a global scale can shed light on the probability distribution of the conclusion we are aiming for.

From the above discussion, it is observed that some of the proposed hybrid models combined several networks to achieve better accuracy for dropout prediction. However, there is still space for improvement, especially incorporating learning patterns and attention mechanisms. Hence, to suggest a model capable of doing that while being efficient and having less computational requirements remains a challenge.

2.2 Models and Existing Techniques

2.2.1 Libraries

To implement all the machine learning and deep learning models, we have taken the help of scikit-learn library. Additionally, for data visualization, we used libraries from matplotlib and seaborn. For loading the dataset and converting columns values to usable data, we have used pandas dataframe and numpy. To solve the class imbalance of our dataset, we have used the oversampling technique SMOTE from SMOTETomek class.

2.2.2 Decision Trees

For classification and regression problems, Decision Tree is one of the most popular machine learning models. It is a tree-like structure with a top-down approach. The initial node of a decision tree is called the root, which represents a feature. The branches from the nodes consist of the different outcomes from that feature and finally, the leaf nodes show the labels, either 0 or 1 in the case of binary classification. The splitting of root nodes is based on some factors such as entropy and information gain. Entropy signifies the purity of a variable and ranges from 0 to 1. The information gained is the net information learned from a feature, and the aim is to maximize it. Greater information gain corresponds to a smaller entropy.[1]

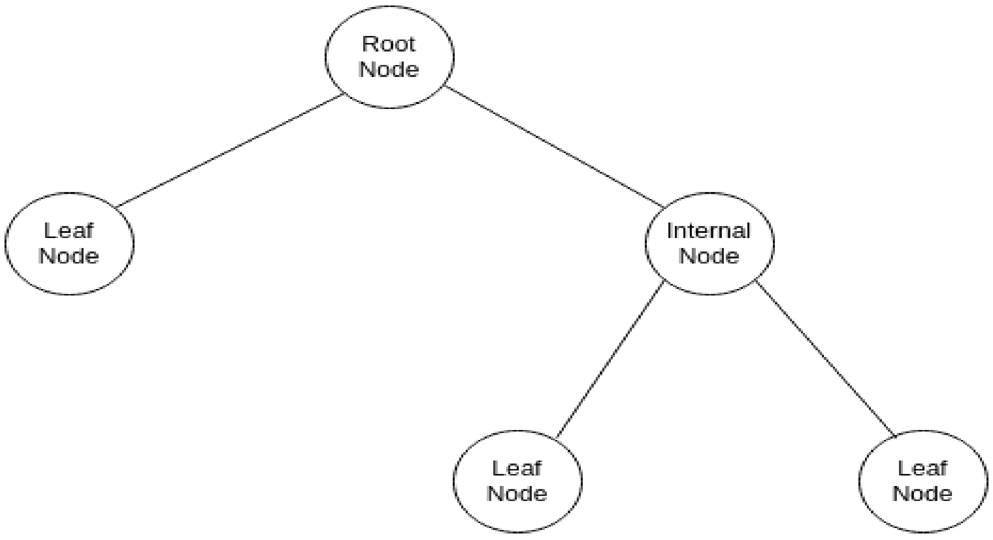


Figure 2.1: Decision Tree [39]

2.2.3 Random Forest Classifier

For categorization problems like ours, a very well-liked machine learning approach called random forest can be applied. It is centred on ensemble learning, which integrates multiple classifiers to enhance the model's performance. Instead of using a single decision tree, it constructs several and averages their predictions. The theory behind this is that since some decision trees may anticipate the incorrect outcome, we will ultimately achieve the best result by averaging with the remaining majority right predictions. In our example, we utilized 600 trees as a balance between performance and computation time; the more trees, the higher the accuracy and the less overfitting occurs.[7]

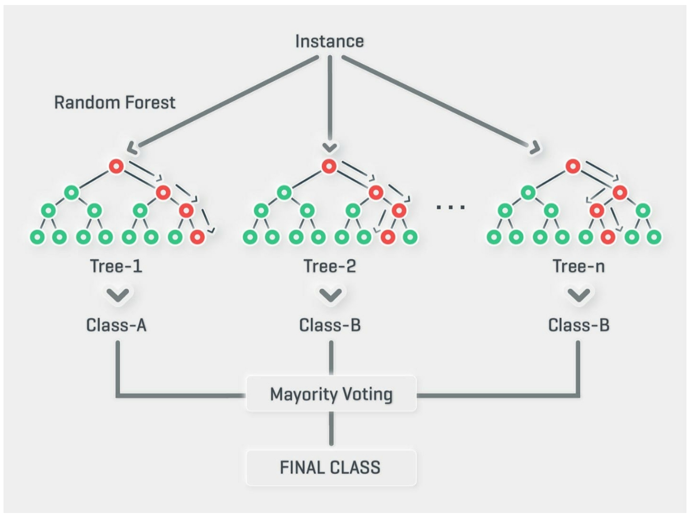


Figure 2.2: Random Forest [37]

2.2.4 Gradient Boosting Classifier

Gradient boosting is a form of boosting machine learning technique that concentrates on building a strong model from weak learners after each iteration. Three things make up gradient boosting: a loss function, a weak learner, and an additive model. In order to comprehend how accurate our model is at categorizing, the loss function is necessary. It reduces the error by selecting a function that favours the weaker hypothesis. Classifying our data is the weak learner's responsibility, but typically it is not accurate enough. However, the additive model re-trains the wrong classification with each iteration, which causes the loss function to shrink with time.[3]

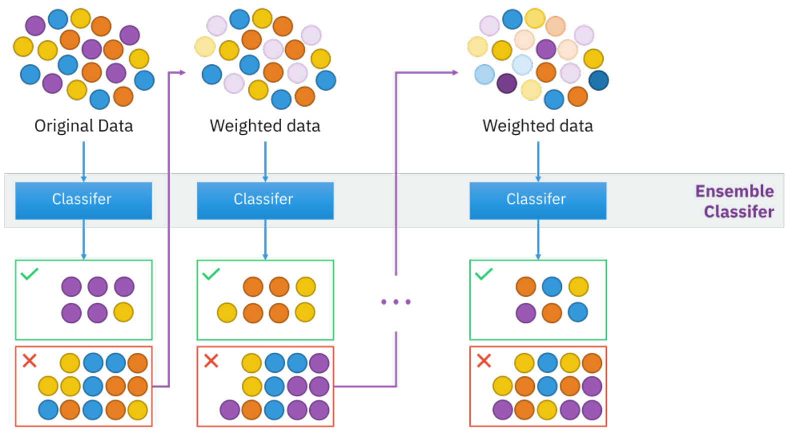


Figure 2.3: Gradient Boosting [28]

2.2.5 Naive Bayes

Naive Bayes algorithm is built on conditional probability. In normal conditional probability, the probability of our output is dependent on all the features combined. To find that, we find the probability of all the features combined dependent on the output. But the calculation is really difficult as it is near impossible to find such a combination of values for each feature for a given output, hence the result comes to zero. To avoid this, Naive Bayes takes a naive approach by treating each feature as independent of one another. So instead of taking all features at a time, naive bayes algorithm takes the probability of one feature dependent on the output, and does this for all features and multiplies it. Now the chance of getting zero is almost negligible. For a binary classification, the probability of both the classes are found and normalized. The class with the higher probability is the output. [9]

2.2.6 Support Vector Machines

Another reliable supervised learning approach that can be applied for classification is support vector machines. The SVM algorithm's goal is to establish the optimum decision boundary for classifying the n-dimensional space. Now, we may categorize fresh data based on which side of the line it lies. The extreme points that aid in forming the boundary are referred to as support vectors, and this line is known as the hyperplane. The margin is the distance between the two extreme examples, and SVM aims to maximize this margin.[2]

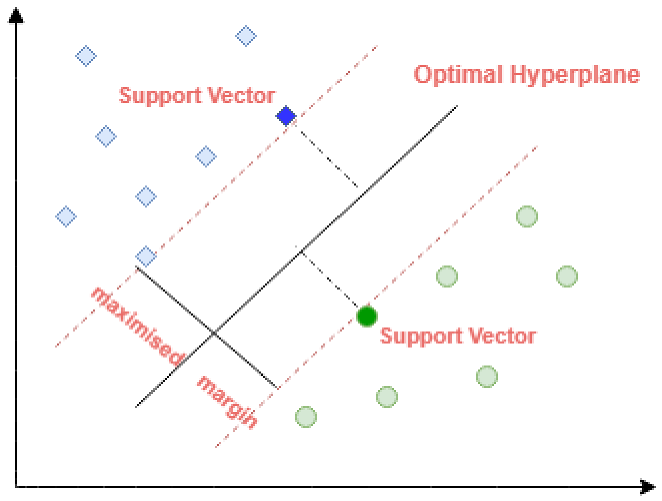


Figure 2.4: Support Vector Machine [41]

2.2.7 Multilayer Perceptron

Multilayer perceptron is the most basic form of deep learning models. Deep Learning is a subset of machine learning which can handle complex and large datasets. MLP consists of one layer of input perceptron, followed by one or more layers of hidden layer perceptron and finally one layer of output perceptron. A perceptron, also known as neuron, is an Artificial Neural Network which behaves similarly to a biological neuron present in the human brain. It takes an input and adds weight to it. Since there are multiple inputs so after adding all the weighted inputs, a bias is also added and this is passed through an activation function. If the generated value from the activation function is more than the threshold value, an output is produced. Most common activation functions used are sigmoid, rectified linear units(ReLU), tanh and softmax. The feedforward network feeds the output to the next layer and this process repeats. By using backpropagation, the weights are recalibrated so that the error is minimized. MLP adds the extra benefit of doing more complex propositional logic like Exclusive OR (XOR) which could not be done in a single layer perceptron. If each neuron of a single layer is connected to all the neurons from the previous layer, it is called a fully connected layer. Multilayer perceptron can be used for both classification and regression problems.[6]

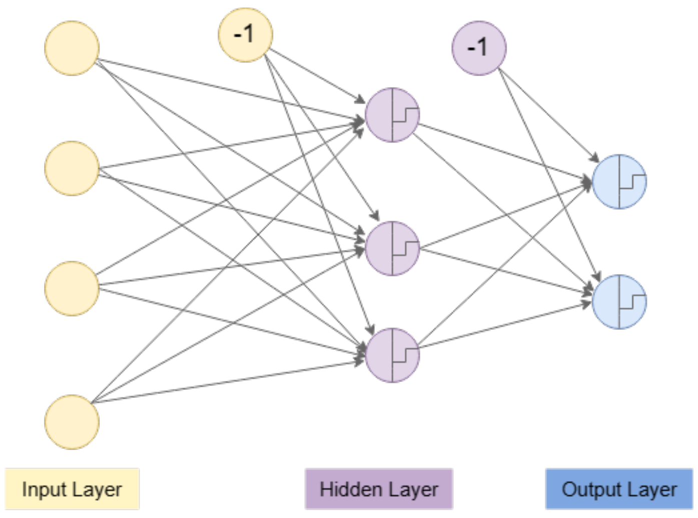


Figure 2.5: Multilayer Perceptron [32]

2.2.8 KNN

The k-nearest neighbours algorithm is a supervised learning classifier that compares new data with similar data points based on a region or neighbourhood. It determines its class according to the class of samples it is closest to. Additionally, the fact that it stores the training dataset rather than using it immediately to learn from makes it a "lazy learner" algorithm. When new data is received, the KNN approach simply refers back to the information it stored during the training phase and assigns it to a category that is very comparable to the new data. For this reason, KNN is sometimes referred to as a memory-based or instance-based learning strategy because it relies heavily on memory to remember all of its training data. KNN algorithm has different ways to calculate the distance to decide it classification such as Manhattan, Euclidean, Minkowski, and Hamming.[5]

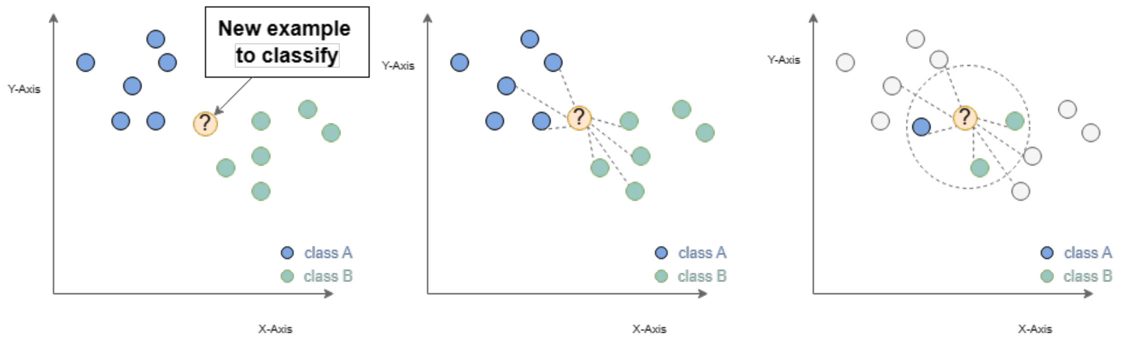


Figure 2.6: K-Nearest Neighbour [40]

2.2.9 Ensemble Learning

Ensemble learning [34] is a powerful technique used to boost the performance of predictions. There are mainly 4 different types of ensembling learning techniques such as bagging, voting classifier, boosting and stacking. Bagging uses the idea of using multiple instances of the same weak learner working in parallel with different subsets of the dataset with replacement. In bagging, the subset of columns can be taken as well. Random Subspace Method is when only a subset of columns is taken with replacement. Random Patches Method is when both column and row subset are taken with replacement. Furthermore, when bootstrap is set to false, there is no replacement in a subset of the dataset, it is called pasting. Random Forest is an example of bagging where the weak learner is a decision tree.

Voting classifiers are similar to bagging but defers with two small exceptions. In the voting classifier, different weak learners are used. Also, the dataset is used as a whole, no subset is taken. The voter classifier votes on the outcome of each base estimator and chooses the outcome with the most votes. The voting criteria is of two types, hard voting and soft voting. Hard voting is calculated based on the predicted output class, whereas soft voting is calculated on the predicted probability.

Thirdly, Boosting merges the weak learners sequentially, retraining the incorrectly classified labels. Boosting weighs the wrong classified data points heavily for the next execution. And finally there is stacking. Stacking is similar to voting. Stacking simply takes the result from the voting classifier and passes it through a final classifier, which increases the performance of the overall model.

Ensemble method not only gives better results but also has been very successful in decreasing the variance resulting in the overfitting to decrease. Thus, the ensemble model performs much better on unseen data compared to the weak learners it uses.

2.2.10 LIME

Local interpretable model agnostic explanations (LIME), work by explaining the black box model by training local surrogates, instead of a global approach. The model internals are hidden from us. The explanations can be validated if we have prior knowledge.

$$explanation(x) = argmin_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{2.1}$$

(2.1) shows the LIME equation

LIME makes a local approximation of the complex model for a specific input. The complex model is represented by f, and g means a simple interpretable model that comes from a set of interpretable models G, containing linear models and their variants. π_x is the proximity of our data point x, which defines the local neighborhood. When we feed different data into a machine learning model, LIME sees what changes happen to the predictions. By randomly shuffling samples and the accompanying predictions of the black box model, LIME creates a new dataset. This new dataset is used by LIME to train an interpretable model by adding weights between the sampled instances and the instance of interest. Minimize the loss function by getting the highest accuracy on the new dataset using a simple linear model. L is the mean-squared error between the label from the complex model and the prediction of the simple model, g. The proximity is added to weigh the loss according to how close the data point is. $\Omega(g)$ is the model complexity, which is kept as simple as possible by using a sparse linear model. They tend to produce as many zero weights as possible. Furthermore, the fewer features, the simpler it is.[14]

2.2.11 SHAP

The concept of SHapley Additive exPlanations (SHAP) comes from cooperative game theory. For instance, if a game is played by a team of 4 members, and they win a certain amount of prize money, how will they split it fairly among themselves? The answer comes from shapley values, which determine the average contribution of each player to the payout. In machine learning, the players can be thought of as features and the payout would be the prediction.

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!}[f_x(z') - f_x(z' \backslash i)] \tag{2.2}$$

(2.2) shows the equation for calculation shapely values

ϕ_i is the shapely value for feature i, where f is the blackbox model and x is the input data point. z' is all possible subsets of features to make sure it accounts for the interaction between individual feature values. x' is the simplified mapped input data that is used in image-related data, where each pixel does not necessarily mean a feature. Then we get the blackbox model output for the subset with and without the feature we are interested in. For instance, in our case, a subset of features could be access and navigate. If we are interested to find the contribution of access, then $f_x(z')$ would give the output for both features together and $f_x(z'/i)$ would give value for only navigate. This difference is known as the marginal value. [16]

The process repeats for all possible combinations of subsets and, additionally, they are weighted according to the number of features present in that subset. M represents the total number of features. The intuition is that the contribution of a feature, say navigate in our case, would be weighted more if the subset already contains many features. It tells us that the addition of this feature gives a strong change even though several features are already included. Now, a feature cannot be removed from a datapoint because the shape would change. SHAP solves this problem by inserting random values from the train dataset into the features we want to exclude. Since random values have no predictive power, the values are shuffled in all subsets.

Chapter 3

Dataset

3.1 Dataset Description

For our experiment, we have used the KDDCUP 2015 dataset derived from XuetangX, one of the largest MOOC platform in China. It is a subset of the bigger XuetangX dataset which was made publicly available. The platform offers a plethora of courses which covers a wide range of subjects. Students have the option to enrol in multiple courses at the same time and enjoy flexible learning schedules. The dataset contains various activities students interact with like navigating the browser chapters, video, sequential access, wiki and average chapter delays. The dataset contains 120542 rows with 141 columns which had 18 prominent features. In the end, we used 12 features after carefully selecting the most important ones, which is discussed later.

Class	Description
label	dropout label of 0 or 1
avg chapter delays	average delays between chapters
parallel enrollments	courses enrolled together
server sequential	accessing servers one after another
wiki	information of particular course
browser problem	course assignment problems
server wiki	fetching wiki data from server
access	accessing other course object other than videos and assignments
browser combined openended	web pages opened indefinitely
browser access	accessing the website
browser video	video content of the courses
class size	total enrollments of a particular course
navigate	navigating other parts of the course
server problem	fetching assignment problems
server chapter	fetching chapter information
server navigate	navigating to the server
browser sequential	websites accessed sequentially
server access	requiring server access from browser

Table 3.1: Dataset Description

3.2 Data Preprocessing

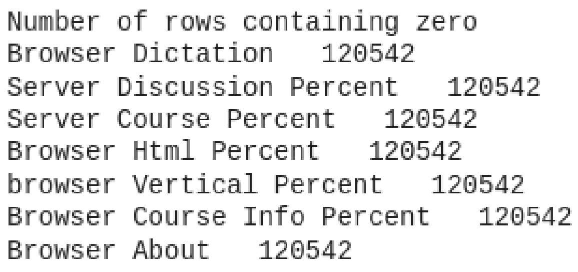
An extremely important phase in machine learning is data preprocessing. Most of the time, raw data cannot be used directly in machine learning models; therefore, it must be cleaned and formatted before being used. Data preparation enhances the accuracy and effectiveness of our models.

3.2.1 Reduction of Features

As mentioned before, the structure of the dataset is not ideal yet to be fed into classification models. It contains about 141 columns in total. Our dataset contains four types of sequential data namely activity day, activity hour, activity week day, session in week which contributes to 67 columns. Details of the sequential columns are shown in table 3.2. We are working with non-sequential models, hence we did not utilize these features. As we are not dealing with that type of data, we dropped those columns. Out of the remaining 74 columns, 55 of them have all zeros, so we have dropped them. We removed the enrolment-id feature from the remaining columns because it does not provide any useful information for training purposes, leaving us with a total of 18 columns to deal with.

Class	Description
act cnt day	Number of student activity on a day of a particular month
act cnt hour	Number of student activity every hour
act cnt week day	Number of student activity on a day of a particular week
session in week	Number of sessions per week

Table 3.2: Description of Sequential features

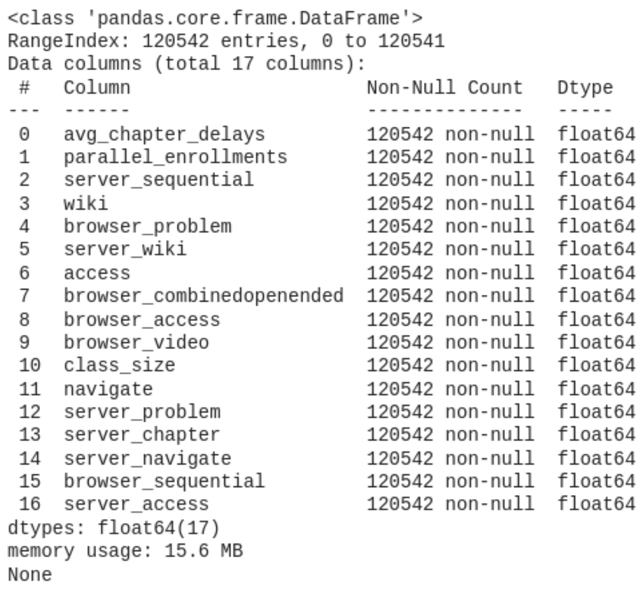


```
Number of rows containing zero
Browser Dictation   120542
Server Discussion Percent   120542
Server Course Percent   120542
Browser Html Percent   120542
browser Vertical Percent   120542
Browser Course Info Percent   120542
Browser About   120542
```

Figure 3.1: Dropping Columns with all zeros

3.2.2 Handling Null and Categorical Values

Our dataset did not contain any null values which we would have to remove. We observed this by counting the non-null values of each column. If all 120542 rows were non-null values, it meant there were no null values. Similarly, by checking the data type of each column, we could say there was no need for encoding categorical values since all the features were either integer or float type.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120542 entries, 0 to 120541
Data columns (total 17 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   avg_chapter_delays        120542 non-null  float64
 1   parallel_enrollments      120542 non-null  float64
 2   server_sequential         120542 non-null  float64
 3   wiki                      120542 non-null  float64
 4   browser_problem           120542 non-null  float64
 5   server_wiki               120542 non-null  float64
 6   access                    120542 non-null  float64
 7   browser_combinedopenended  120542 non-null  float64
 8   browser_access            120542 non-null  float64
 9   browser_video             120542 non-null  float64
 10  class_size                120542 non-null  float64
 11  navigate                  120542 non-null  float64
 12  server_problem            120542 non-null  float64
 13  server_chapter            120542 non-null  float64
 14  server_navigate           120542 non-null  float64
 15  browser_sequential        120542 non-null  float64
 16  server_access             120542 non-null  float64
dtypes: float64(17)
memory usage: 15.6 MB
None
```

Figure 3.2: Checking Null and Categorical Values

3.2.3 Recursive Feature Elimination

Feature	Rank	Drop
avg chapter delays	1	No
server sequential	1	No
browser problem	1	No
access	1	No
browser access	1	No
browser video	1	No
class size	1	No
navigate	1	No
server problem	1	No
server chapter	1	No
server navigate	1	No
server access	1	No
parallel enrollments	2	Yes
browser sequential	2	Yes
wiki	3	Yes
server wiki	3	Yes
browser combined openended	3	Yes

Table 3.3: Recursive Feature Elimination

Recursive Feature elimination uses an estimator, in our case it is the Random Forest Classifier, and assigns weights to the available features. Initially it starts with 17 features and after every iteration it drops the least important features. Hence, by working recursively, it is considering a smaller set of features in every execution. The total number of iterations is 3 in our case. So in this experiment we are left with 12 features after 3 recursions. We experimented this process using different estimators and found that Random Forest estimators selects the features that perform the best.

3.2.4 Normalization

Class	Before Normalization	After Normalization
avg chapter delays	(0.0, 5.0)	(-0.44, 25.73)
server navigate	(0,649)	(-0.65, 49.38)
server sequential	(0,3619)	(-0.43, 121.95)
access	(0,3659)	(-0.47, 66.76)
class size	(118,7310)	(-1.12, 2.28)

Table 3.4: Normalization

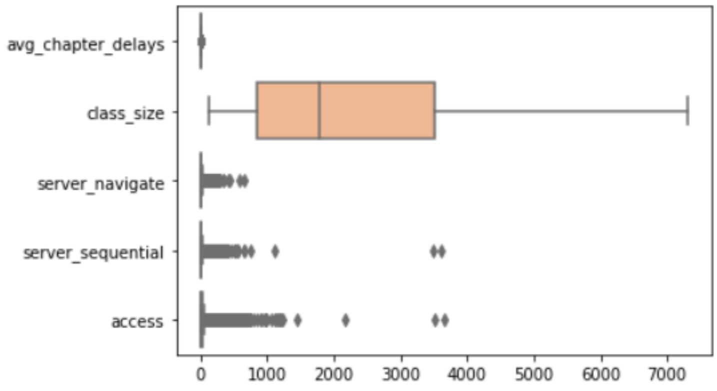


Figure 3.3: Before Applying Standard Scalar

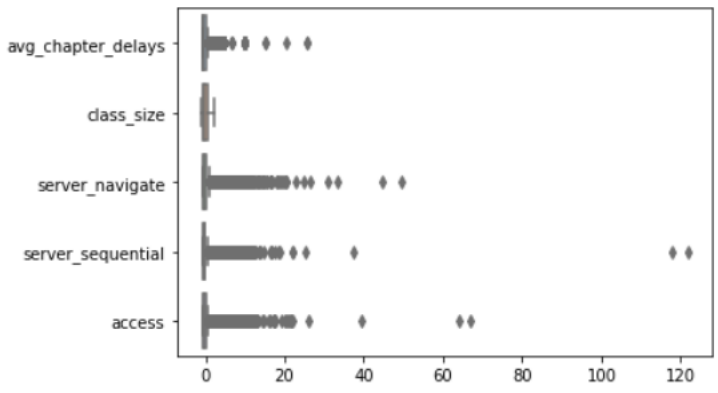


Figure 3.4: After Applying Standard Scalar

The key idea of normalizing is to modify the numeric values of a column to make it uniform. The structure of the data becomes a normal distribution with even number of data above and below the mean. For example, average chapter delays column has values ranging from 0.0 to 5.0 whereas access has values ranging from 0 to 3659, so this difference in range needs to be handled. We have used standard scalar technique to normalize all our data. It scales the values by subtracting the mean from it and then scaling it to the unit variance. Thus, it also contains negative values and the final range is from -1 to +1 with the mean at 0.

There are other two scalars, namely min-max scalar and robust scalar. We did not use min-max scalar for our dataset because it reduces the range of the data as the lowest value is 0 and highest is 1, so there is some data loss. Features like access and class size has a wider range in as shown in table 3.4, so min-max scalar will hamper these column values. In our MLP model, this scalar impacts the most as the 0 values are treated as null values and hence it could affect the learning process.

On the other hand, robust scalar works by ignoring the extreme outliers for each feature column. Furthermore, this scalar does not aggressively scale compared to standard scalar, since it only scales using data from the 25th percentile to the 75th percentile. However, in our dataset there is very little outliers to remove and as a result this scalar is not suitable to use since it does not normalize the values in a wider range like the previous two explained earlier.

From figure 3.3, we can see the difference in range of our features clearly. The class size feature has a far wider range compared to columns like avg chapter delays and server-navigate. So to normalize the range of data in order to make all of them uniform with each other, we applied standard scaling. The figure 3.4 shows the effect of scaling and how the data is distributed evenly across all the features.

3.2.5 Train-Validation-Test Splitting

Finally, we divided our dataset into train, validation and test sets in proportions of 60-20-20 respectively.

Train: Train set is used to fit our models so that they can learn the parameters of the dataset.

Validation: The validation set is used to tune the hyperparameters of our models. Each set of hyperparameters is run on the validation set to find out which combination of parameters gives the best results.

Test: Finally, the test set is used to evaluate the performance of our models at the end. It only runs once, compared to the validation set. We have also stratified the splitting process on the basis of the target variable. This ensures that the ratio of 0s to 1s in the train and test sets is the same.

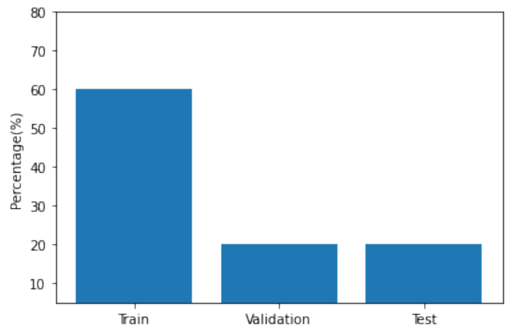


Figure 3.5: Train-Validation-Test Split

Chapter 4

Proposed Model

4.1 Different Ensemble Models with XAI

Our approach in predicting dropout takes the help of different ensemble models. Taking the majority votes from five of our best performing models, it gives a more robust and accurate outcome. The concept of voting is based on the assumption that some models might produce inaccurate results. However, the bulk of the models would have correctly anticipated the outcome, therefore the final result would be the correct one if we had cast our votes based on each model's prediction. We have implemented both hard and soft voting as well as stacking; hard voting which calculates the result based on the predicted output class, compared to using probabilities in soft voting. Stacking works similarly like hard voting with an additional classifier at the end for the final prediction.

At first, we applied all the necessary pre-processing techniques to clean and normalize our dataset. Next, we split our dataset into train, validation and test sets. The training set is then fed into our voting classifier containing all the models. The voting classifier creates copies of the training set for each model. Moreover, we have used an odd number of classifiers to avoid any indecision in case of equal number of votes from both classes. For example, if we had four models, out of which two models gave the outcome as 0 and the other two as 1. In this case, we will not get a concrete outcome to confidently say which one is the correct result. Using an odd number of models solves this issue by always giving a definite result, as there is an unequal number of 1s and 0s.

For the voting estimators, we have used gradient boosting, random forest, decision tree, K-nearest neighbour and multilayer perceptron as these models gave the best performance. For the final estimator in stacking, we selected gradient boosting as the final estimator as it performed the best compared to using other final estimators.

Finally, we explained our model's decisions of the test samples using explainable artificial intelligence (XAI). The libraries we used for this purpose are LIME and SHAP.

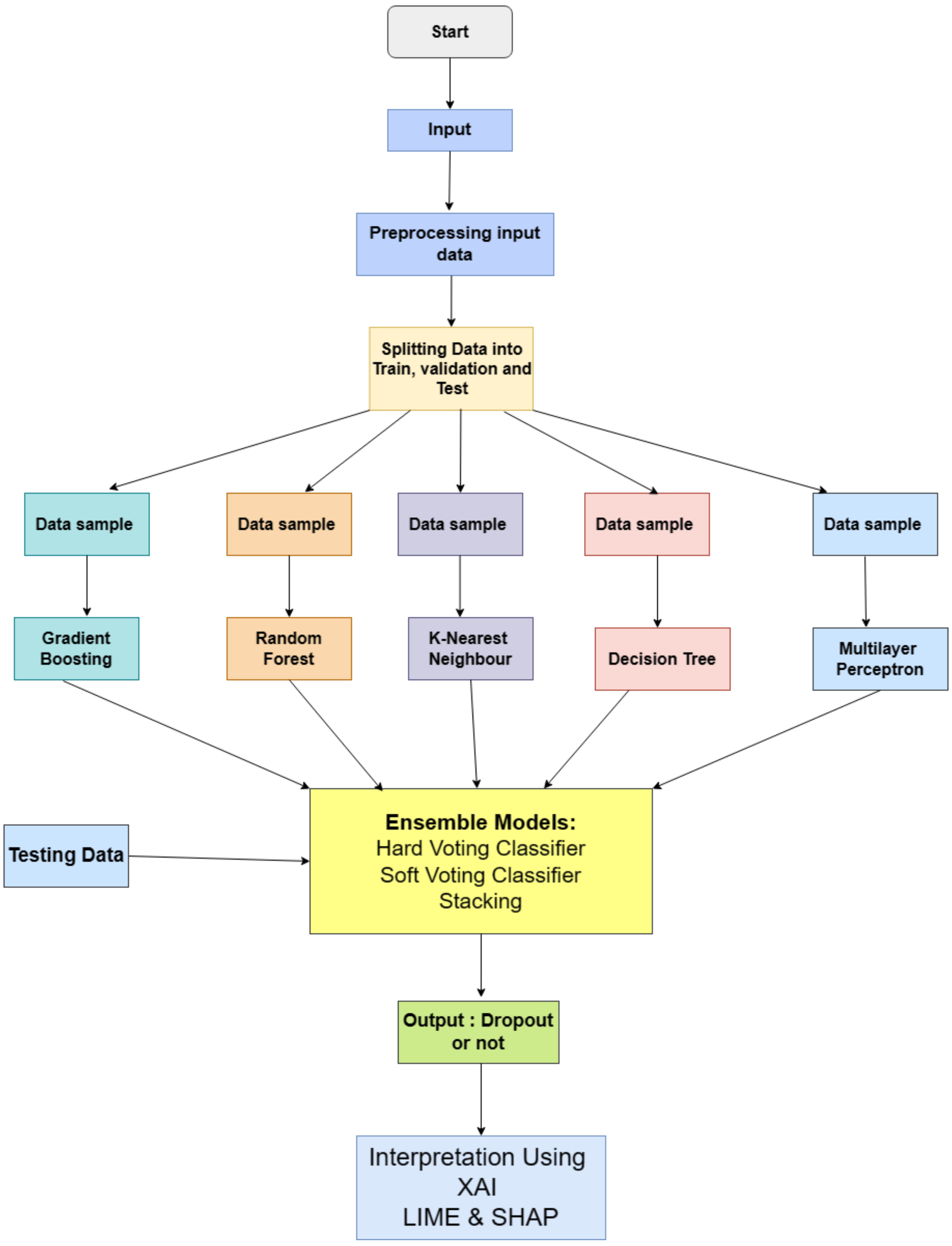


Figure 4.1: Proposed Model

Chapter 5

Experiment and result analysis

To access the performance of our models, we have used the following metrics: recall, precision, f1-score, accuracy derived from the confusion matrix. In addition, we also used ROC and AUC (area under the curve) as they have been proved to be highly effective in past papers to judge the KDDCUP 2015 dataset. The parameters used from the confusion matrix are TP (True positive), TN (True negative), FP (False positive) and FN (False negative).

TP signifies the correctly identified dropout students.

TN implies the correctly identified non-dropouts.

FP indicates the wrongly identified dropouts.

FN shows the wrongly identified non-dropouts.

The following equations were used to calculate the metrics.

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

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

$$F1 - score = \frac{2TP}{2TP + FP + FN}$$

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

Recall is the fraction of actual positives actually labelled positive. In our research, it is the fraction of total dropouts which are actually labelled as dropouts. According to the recall formulae, to get a higher recall, the false negative has to be lower. Meaning to get a higher recall, the actual dropout being labelled as non-dropouts must be lower. Recall is used as a primary metric in cancer research, as having low false negatives in cancer research is vital. We do not want to miss any cancer patients while predicting.

Precision on the other hand is the fraction labelled positive which is actually positive. In our research, it is the fraction of labelled dropouts which are actually dropouts. According to the precision formulae, to get a higher precision, the false positive has to be lower. Meaning to get a higher recall, the non-dropout being labelled as dropout must be lower. Precision is widely used in detecting faulty machinery, as having low false positives is vital.

F1-score is the harmonic mean of the recall and precision. Here we are using harmonic mean and not actual mean or average. The reason is that the harmonic mean is greatly influenced by the lower value, while the average will give the midpoint between precision and recall. If we have a model where the recall is 20 and precision is 90, average will give 55. However, the harmonic mean will give 32.73. Here it is evident that harmonic mean gives a better representation of our model compared to mean or average.

In majority research, accuracy has been used as the primary measure to determine the performance of the model. However, accuracy is not a good measure when it comes to highly unbalanced dataset. Suppose we have a dumb model which predicts a dog whenever an image is shown to it. Now let's say in our dataset 97% are dog images and the rest are cat images. If we use this model to classify, then the accuracy comes to 97% as the model predicted the correct label for 97% of the dataset.However, this is a false representation of the model, as the model did not perform well at all. The same model will have an accuracy of 10% if 10% of the images in the testing dataset were dog images. So for highly imbalanced dataset we use recall, precision and f1-score as our primary evaluation metric.

The MOOC dataset we used in this research roughly contains four times as many dropouts as non-dropouts. Here we are trying to predict dropouts. So just like cancer research, here we need to have low false negatives. It's vital we do not label a dropout as a non-dropout. So recall is our primary metric. In addition, we used precision and f1-score to get an overall idea of non-dropout as well.

In binary classification, there is only one value of accuracy for both the classes. However, the value of precision, recall and f1-score can be different for two classes. In this research, our main focus is to predict the dropout, so our primary point of focus is the dropout class. Hence, for precision, recall and f1-score we took the values from the dropout class.

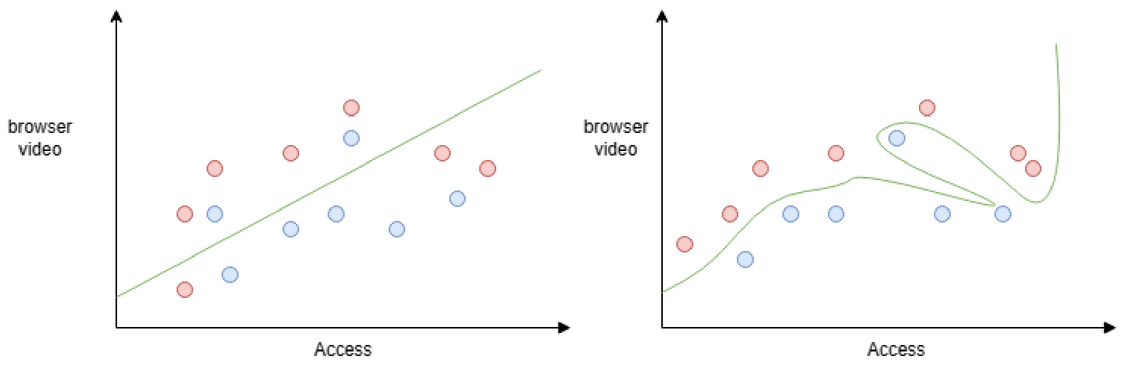


Figure 5.1: Underfitting and Overfitting

In underfitting the model is way too simple. As a result, it can not find enough connections between features and the target label. This means the accuracy is low for both training and testing phase. On the left side of the figure 5.1, we can see there is a straight line and there are red and blue data points on either side. This shows few data points were wrongly classified. And hence the accuracy is low. Underfitting results in the model to have high bias. There are few ways to tackle underfitting in the model. One way is to make the model complex by increasing the degree of polynomials. For example, if the model is quadratic, we can make it cubic. Another way can be to increase the number of features so that the model can learn more parameters during the testing phase. Finally, we can reduce the regularization so that the model has greater variance.[26]

Overfitting is the exact opposite of underfitting. Overfitting is the tendency of the model to learn the training set too well. Overfitting makes the machine learning model too complex. As we can see from the figure 5.1, the right graph shows the line separating the blue datapoint to the red datapoint in the training set. If we provide a testing set, the new datapoint can fall on the wrong side and will be mislabelled. This results in the model having low accuracy on the unseen or training data. Overfitting results in the model to have high variance. There are multiple ways we can handle overfitting. One way is to make the model simple by reducing the degree of polynomials. For example, if the model is quadratic, we have to make it linear. Another way is to reduce the number of features so that the number of variables in the equation decreases and the model has to learn fewer parameters during the training phase. Finally, we can increase regularization to decrease the variance. The larger the difference between training and testing accuracy, the larger is the overfitting.[26]

We performed hyperparameter tuning using nested for loops with each parameter containing its possible values in a list. We tested every combination of parameters and choose the set of hyperparameters that gave the highest recall and F1-score.

5.1 Decision Trees

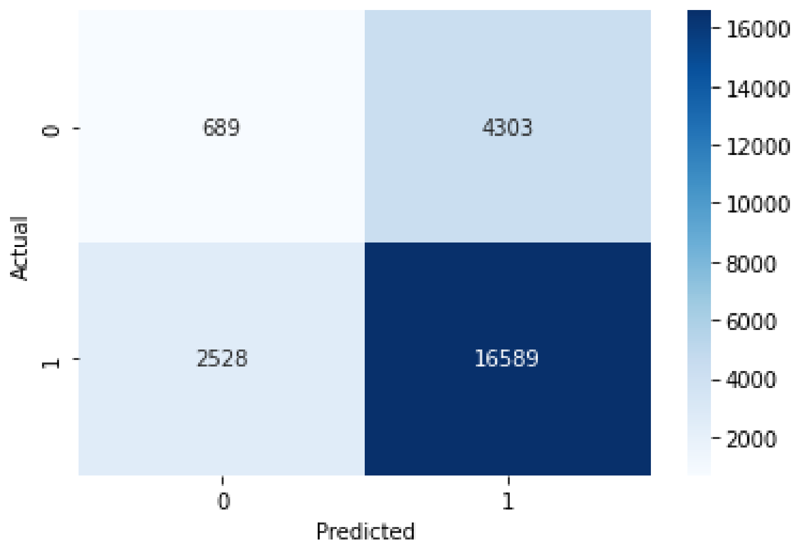


Figure 5.2: Confusion Matrix of Decision Tree

Set	Recall	Precision	F1-score	Accuracy
Train	95.262%	88.272%	91.634%	86.208%
Validation	95.616%	87.816%	91.551%	86.005%
Test	94.947%	87.932%	91.305%	85.660%

Table 5.1: Performance Metrics for Decision Tree

Hyperparameter	Possible Values	Chosen Value
Criterion	[gini, entropy]	gini
Max Depth	[2, 3, 5, 10, 20]	10
Min Sample Leaves	[5, 10, 20, 50, 100, 150]	100
Max features	[auto, sqrt, log2]	auto

Table 5.2: Hyperparameter for Decision Tree

The decision tree is designed to halt splitting when it gets a leaf with pure samples. This means either all the samples are dropouts or non-dropouts. As max-depth is 10, so the decision tree will stop splitting after it reaches a depth of 10 irrespective of it getting a pure leaf or not. When max-features is set to 'auto' then square root of available features are taken into consideration for splitting. 'Gini' is used as a criterion for splitting, which is computationally less demanding. Min-sample-leaf = 100 means that if there are less than 100 samples in an internal node, the node is not further divided and hence becomes a leaf node.

5.2 Random Forest

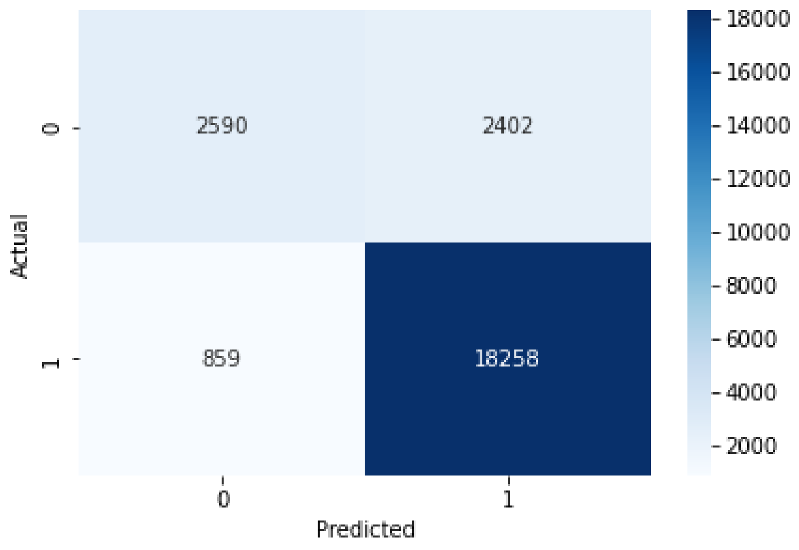


Figure 5.3: Confusion Matrix of Random Forest

Set	Recall	Precision	F1-score	Accuracy
Train	96.764%	89.352%	92.910%	88.290%
Validation	95.496%	88.390%	91.806%	86.482%
Test	95.360%	88.089%	91.580%	86.096%

Table 5.3: Performance Metrics for Random Forest

Hyperparameter	Possible Values	Chosen Value
N-estimators	[200,300,400,500]	400
Criterion	[gini, entropy]	gini
Max Depth	[2, 10, 20, 50,100,150]	10

Table 5.4: Hyperparameter for Random Forest

Random forest is a bagging classifier consisting of decision trees. Here we have used 400 decision trees. Normally, the splitting continues until we get a leaf with pure samples, meaning either all of them are dropouts or non-dropouts. But by limiting the max-depth to 10, the decision tree will stop splitting when it reaches a depth of 10. Both of these will prevent over-fitting. We found ‘gini‘ to be the best criterion. This is set as default. It works as well as entropy but is computationally less demanding, so overall performance improves. Although Random Forest performs better than decision tree as expected, random forest overfitting more compared to the decision tree. This is likely because the random forest used overlapping subsets of the dataset for each tree, which increases the variance. This is due to the fact that bootstrap is set true by default, and we did not change it.

5.3 Gradient Boosting

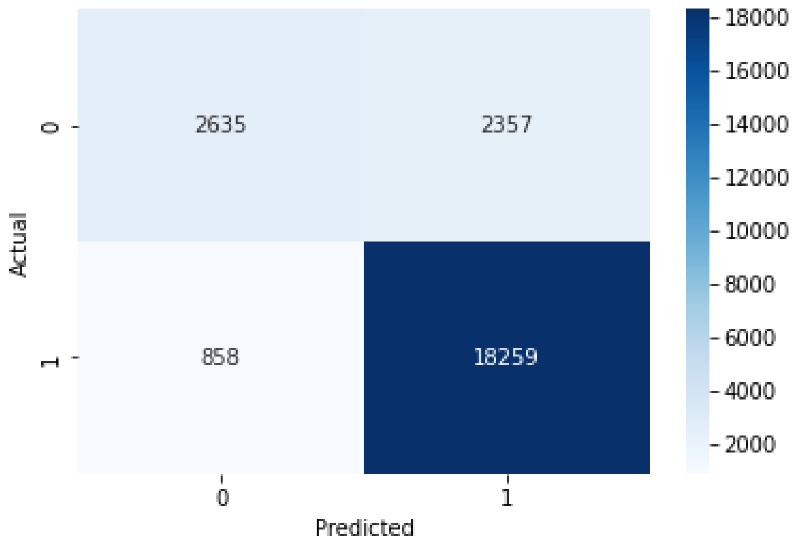


Figure 5.4: Confusion Matrix of Gradient Boosting

Set	Recall	Precision	F1-score	Accuracy
Train	97.070%	86.558%	91.513%	85.723%
Validation	96.511%	86.086%	91.001%	84.865%
Test	96.166%	85.943%	90.767%	84.487%

Table 5.5: Performance Metrics for Gradient Boosting

Hyperparameter	Possible Values	Chosen Value
N-estimators	[200,220,240,250,270,300,320,340,350]	200
Learning Rate	[0.1,0.5,1,1.5,2]	1.5

Table 5.6: Hyperparameter for Gradient Boosting

The n-estimators refers to the number of trees used in the model.Each individual tree is generated to reduce the errors produced in each iteration. Generally, the more trees the better the model will learn the data, however, a high number of trees slows down the training phase considerably. So we need to find a balance between performance and training time, which after several tuning we found to be 200. Another hyperparameter that is used in gradient boosting is the learning rate, which determines how quickly or slowly the model fits the training data. If the models fit quickly, it will overfit the data, so we need to set a learning rate that allows for good learning without overfitting. In our case, a learning rate of 1.5 worked the best.

5.4 Gaussian Naive Bayes

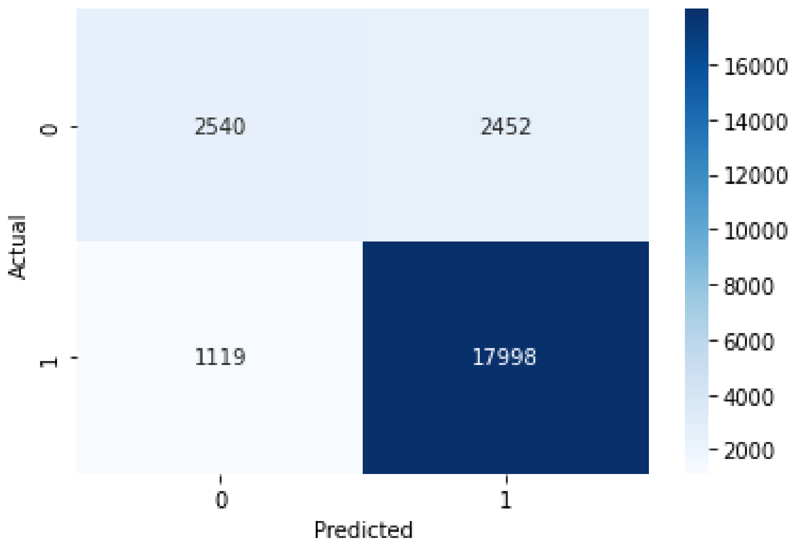


Figure 5.5: Confusion Matrix of Naive Bayes

Set	Recall	Precision	F1-score	Accuracy
Train	94.504%	87.619%	90.931%	85.053%
Validation	94.455%	87.753%	90.981%	85.151%
Test	94.173%	87.380%	90.650%	84.595%

Table 5.7: Performance Metrics for Naive Bayes

Here we did not tune the hyperparameters, as the available parameters give the best results. Naive Bayes algorithm assumes that the features are independent of one another and ignore any correlation among features while fitting the model. The dataset we have used is from the real world MOOC platform, so it reflects the real world phenomenon. In the real world, the features are indeed correlated with one another and thus while fitting the model this has to be taken into consideration. All other models that have been used take the correlation into account, so it significantly outperforms the Naive Bayes classifier. A Gaussian Naive Bayes model just takes the data and converts it into a normal or Gaussian distribution before beginning the training phase. Our dataset that is being given to the model to train is already in normal distribution from our preprocessing phase.

5.5 Support Vector Machine

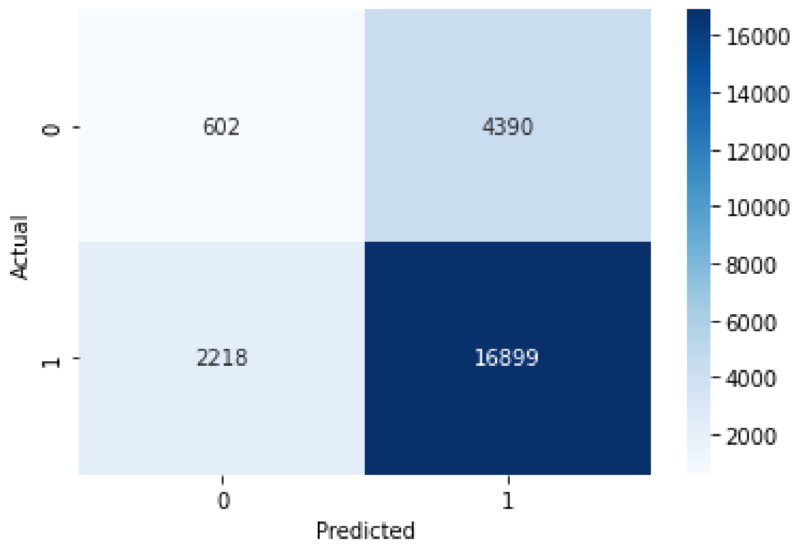


Figure 5.6: Confusion Matrix of SVM

Set	Recall	Precision	F1-score	Accuracy
Train	96.607%	86.832%	91.459%	85.693%
Validation	96.600%	86.773%	91.423%	85.628%
Test	96.338%	86.526%	91.168%	85.200%

Table 5.8: Performance Metrics for Support Vector Machine

Hyperparameter	Possible Values	Chosen Value
Kernel	[rbf, sigmoid, linear]	linear
C-Value	[0.1,1,100]	0.1

Table 5.9: Hyperparameter for Support Vector Machine

SVM uses a decision boundary to separate the binary classes. In this case, one class is dropout and the other is dropout. There are different types of decision boundaries, such as mentioned in the table 5.9. After hypertunning we got a linear decision boundary as our best choice. Linear decision boundary draws a straight line and divides the two classes on either side. C is the regularization parameter. The degree of regularization has an inverse relationship with the C value. Hence, as the C value increases, the regularization decreases and vice versa. Here we used the lowest possible value of C, hence the maximum possible regularization. This means we are ignoring complexity in the model to ensure that we do not overfit.

5.6 Multilayer Perceptron

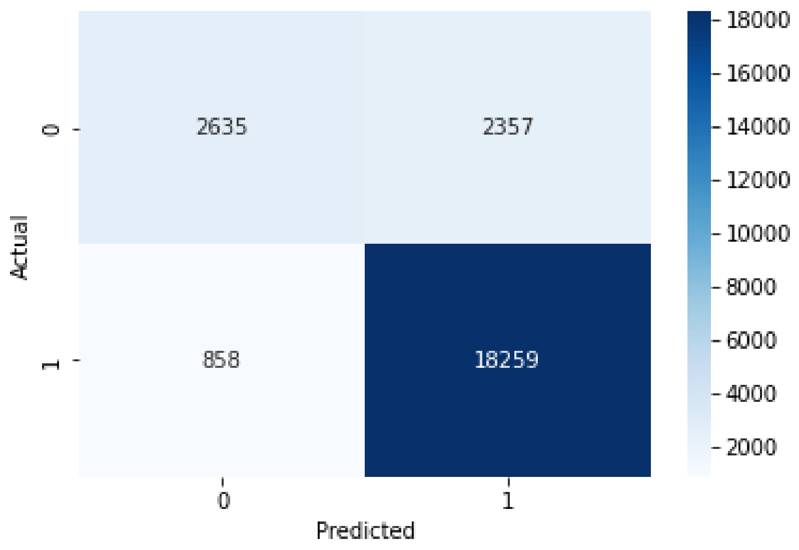


Figure 5.7: Confusion Matrix of Multilayer Perceptron

Set	Recall	Precision	F1-score	Accuracy
Train	95.438%	88.547%	91.864%	86.595%
Validation	95.172%	88.643%	91.792%	86.503%
Test	95.386%	88.305%	91.709%	86.055%

Table 5.10: Performance Metrics for Multilayer Perceptron

Hyperparameter	Possible Values	Chosen Value
iteration	[1,3,5,10,15,20,50,100,150]	20
Activation function	[relu, tanh, logistic]	relu

Table 5.11: Hyperparameter for Multilayer Perceptron

Multilayer perceptron is the most widely used deep learning algorithm. There are multiple different activation functions to choose from, but after hypertuning we found out that 'relu' gives the best result. Relu is short for Rectified Linear Unit. This activation function is a piecewise linear function which does not have any output if the summation of dot product between weight and sum is zero. But if the summation is positive, relu gives the same exact positive out, as it has a linear relation with positive input into the function. As discussed earlier, after a run, MLP calculates the error and performs backpropagation to adjust weights. Then it runs again and again calculates the error, followed by backpropagation to adjust weights. This is continued a total of 20 times to get the best possible result with the least error.

5.7 K-Nearest Neighbour

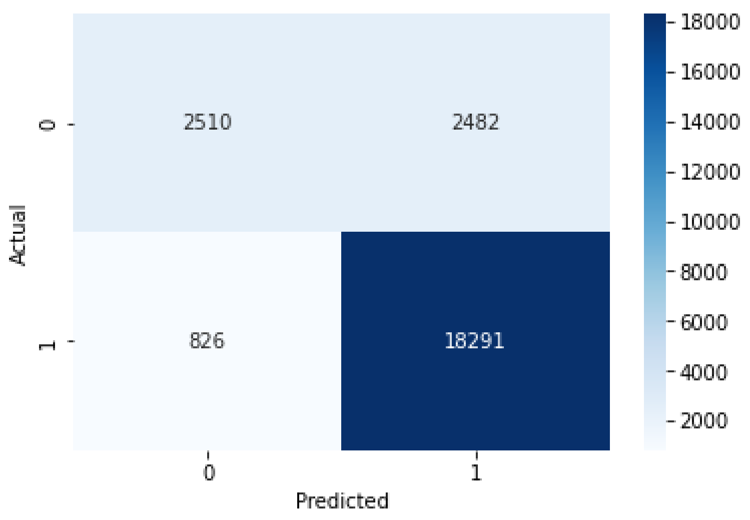


Figure 5.8: Confusion Matrix of K-nearest neighbour

Set	Recall	Precision	F1-score	Accuracy
Train	95.927%	88.193%	91.897%	86.587%
Validation	94.638%	88.086%	91.245%	85.599%
Test	95.308%	87.858%	91.431%	85.835%

Table 5.12: Performance Metrics for K-nearest neighbour

Hyperparameter	Possible Values	Chosen Value
n-neighbours	[13,15,17,19,27,32,34,35,36,38,45,50,55,60,65,70]	35
Weights	[uniform, distance]	uniform

Table 5.13: Hyperparameter for K-nearest neighbour

The n-neighbours value of 35 means the KNN algorithm takes the nearest 35 data points into consideration and votes to find the majority class. If the number of dropouts is greater than non-dropouts, then it predicts dropout and vice versa. It selected 35 and not any surrounding even number like 34 or 36, because in binary classification, an odd number of datapoints acts as tiebreaker. If weights = "distance" parameter was used, then the further data points will have less influence on the prediction, which is not particularly advantageous for binary classification. KNN works best where all the features are scaled to the same level. That is the reason we have used Standard Scalar as our normalization technique. So weights = "uniform" is chosen.

5.8 Comparison between Number of Features used

Number of Features	Recall
17	95.737%
13	96.995%
12	**97.636%**
11	97.113%
10	93.875%

Table 5.14: Performance Comparison between Number of features

As we can see from the table 5.14, using a different number of features impacts the performance. From our initial 17 features, we reduced it further by using recursive feature elimination that ranks the important features. We have to provide the number of features it reduces it to, so after some experimentation we found 12 features to have the best result as it gave a 97.636% recall score. Reducing the features even more, reduces the performance. This also proves that using 12 features reduces overfitting, as the performance improves compared to 17.

5.9 Summary of Individual Models

Model	Recall	Precision	F1-score	Accuracy
DTC	94.947%	87.932%	91.305%	85.660%
RFC	95.360%	88.089%	91.580%	86.096%
GBM	96.166%	85.943%	90.767%	84.487%
GNB	94.173%	87.380%	90.650%	84.595%
SVM	96.338%	86.526%	91.168%	85.200%
MLP	95.386%	88.305%	91.709%	86.055%
KNN	95.308%	87.858%	91.431%	85.835%

Table 5.15: Summarized Performance Metrics for all models

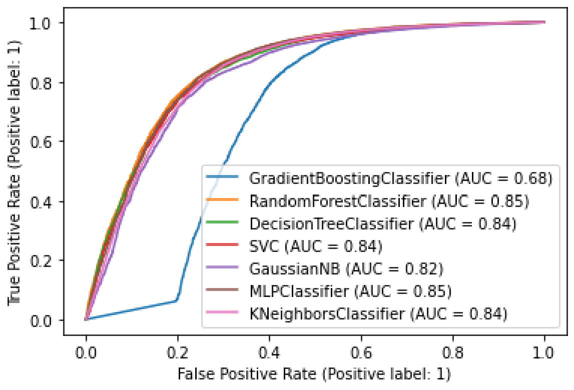


Figure 5.9: ROC Curve of all the models

Overall, most of the machine learning models performed well, with a few performing a little lower. The above table 5.15 shows that random forest performs really well in recall score compared to decision tree. Since it is a bagging classifier, it uses multiple weak learners, which improves its performance. On the other hand, naive bayes performs the worst due to its assumption of features being independent and not taking into account the correlation between them. Although gradient boosting scored lower in precision and F1-score, it had a higher recall metric compared to others. Since recall is our primary focus, we chose gbm as one of the voters over svm. The reason behind is that svm gave much poorer performance when used as a voter, and it takes significantly more time to run. SVM tries to fit the best hyperplane with the training data, however, from the figure 5.11, we can see that our data has a lot of overlapping samples which makes it very difficult for SVM to optimize the hyperplane. Hence, it takes a lot of time to run. All of our models gave a higher recall score than precision. This is due to our dataset being highly imbalanced, with dropouts samples almost 4 times more than non-dropouts. So there is a tendency for our models to predict non-dropouts as dropouts, giving a higher number of false positives. With a small sample size for non-dropouts, the models cannot learn it properly, as a result the precision score falls compared to recall.

5.10 Summary of Different Ensemble Models

Model	Recall	Precision	F1-score	Accuracy
Majority Voting (Soft)	96.453%	87.128%	91.554%	85.889%
Majority Voting (Hard)	95.611%	87.917%	91.603%	86.101%
Stacking	97.636%	81.847%	89.048%	80.957%

Table 5.16: Summarized Performance Metrics for different Ensemble Models

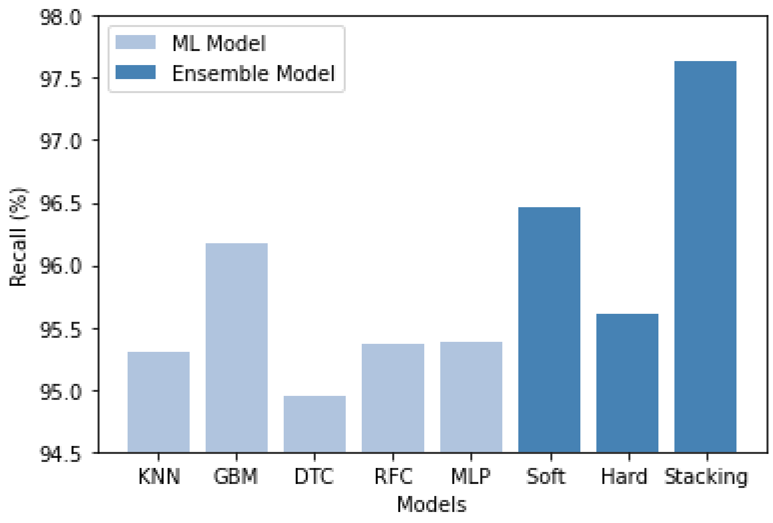


Figure 5.10: Recall comparison between Ensemble and baseline models

Our proposed model uses the different ensemble learning techniques to boost the performance. We have done this by selecting five out of the seven models and used them as voting estimators. We dropped Naive bayes and SVM, since they performed the worst overall in our ensemble models. The objective was to filter out the wrong predictions of some models, with the majority correct predictions made by the rest. Looking at the results, we can say that we have achieved the highest recall score in stacking and highest F1-score in hard majority voting. Since we have an imbalanced dataset, these metrics are important as we are trying to predict dropouts only which label 1 in our case.

Soft voting gave a higher recall score than hard voting with 96.453% and hard voting gave us a higher precision and F1-score than soft voting. Soft voting gives more weight to highly confident votes, which is why it can predict actual dropouts as dropouts giving a better recall than hard voting.

Stacking uses two levels of estimators for prediction. Firstly, it uses 5 base models as voters, similar to hard voting. Then it uses an additional estimator in the second layer for the final prediction. We used gradient boosting as the final estimator. From the table 5.15, we can see that gbm has the significantly higher recall score than other models, making it an ideal choice for the final estimator in the second layer. As a result, stacking produced the highest recall score of 97.636% and its lower precision and F1-score is due to the lower score of gbm in those metrics.

In our research, recall is our target, so stacking would be the best model, however, other papers gave importance to precision scores. If precision and F1-score is needed, we will choose majority hard voting classifier as the model, but for recall, stacking would the best choice.

5.11 Comparison with different Number of Voters

Number of Voters	Recall
4	94.879%
5	**95.611%**
6	95.350%

Table 5.17: Performance Comparison with different number of estimators

We experimented our ensemble models with varying number of estimators to see the difference in performance. We found that using an odd number of voters gives a better recall score, as shown in 5.17. This is due to the fact that odd voters act as a tiebreaker, whereas using an even number of voters would create situations where the number of votes from both classes are equal.

5.12 Comparison of Overfitting

Model	Train-Test Difference
Average of 5 models	0.662%
Soft Voting	0.500%
Stacking	0.380%

Table 5.18: Performance Comparison of Overfitting

The difference in train and test in recall metric determines the amount of overfitting, a higher percentage means more overfitting and vice-versa. Not only did soft voting and stacking perform better compared to our baseline machine learning models, it also handled overfitting well by reducing the variance. From the table 5.18, we can see that the difference between the train and test set are lower in soft voting and stacking compared to the average of 5 baseline models.

5.13 Comparison with and without Oversampling

One of the major problems with our dataset is the highly imbalanced labels of dropout. We have 95581 labelled as 1 and 24961 labelled as 0, so almost four times. So we tried to use an oversampling technique known as SMOTE (Synthetic Minority Oversampling Technique) to balance the dataset and compare how it performs against our initial approach.

SMOTE works by randomly selecting a sample, then finding the k-nearest neighbours of it and choosing one of them. Then it creates a new synthetic sample based on the average of the selected samples. This increases the number of minority class samples so that we have a 1:1 ratio of labels and enables our models to learn both classes with equal importance. This approach also avoids duplication of data, which some oversampling methods use.[27]

Model	Oversampling	Recall	Precision	F1-score	Accuracy
Majority Voting (Hard)	YES	91.215%	89.401%	89.977%	89.955%
Majority Voting (Hard)	NO	95.611%	87.917%	91.603%	86.101%
Stacking	NO	97.636%	81.847%	89.048%	80.957%

Table 5.19: Performance comparison with and without Oversampling

Using oversampling has performed worse compared to keeping the dataset imbalanced as we can see from the lower recall score which means it was not able to correctly identify more true positives and instead identified more false negatives. It means it did not correctly identify dropouts as actual dropouts, rather it identified it as non-dropouts. We can further explore this by pairplotting some of the features of our dataset. From the figure 5.11, we can see that there is a lot of overlapping between the two classes, which makes it tough for SMOTE to select the correct neighbours of the same class. The synthetic data that is being generated is usually placed around similar datapoints which creates further overlapping problems. Furthermore, since it creates artificial data, it manipulates the original data which does not depict a real world scenario.

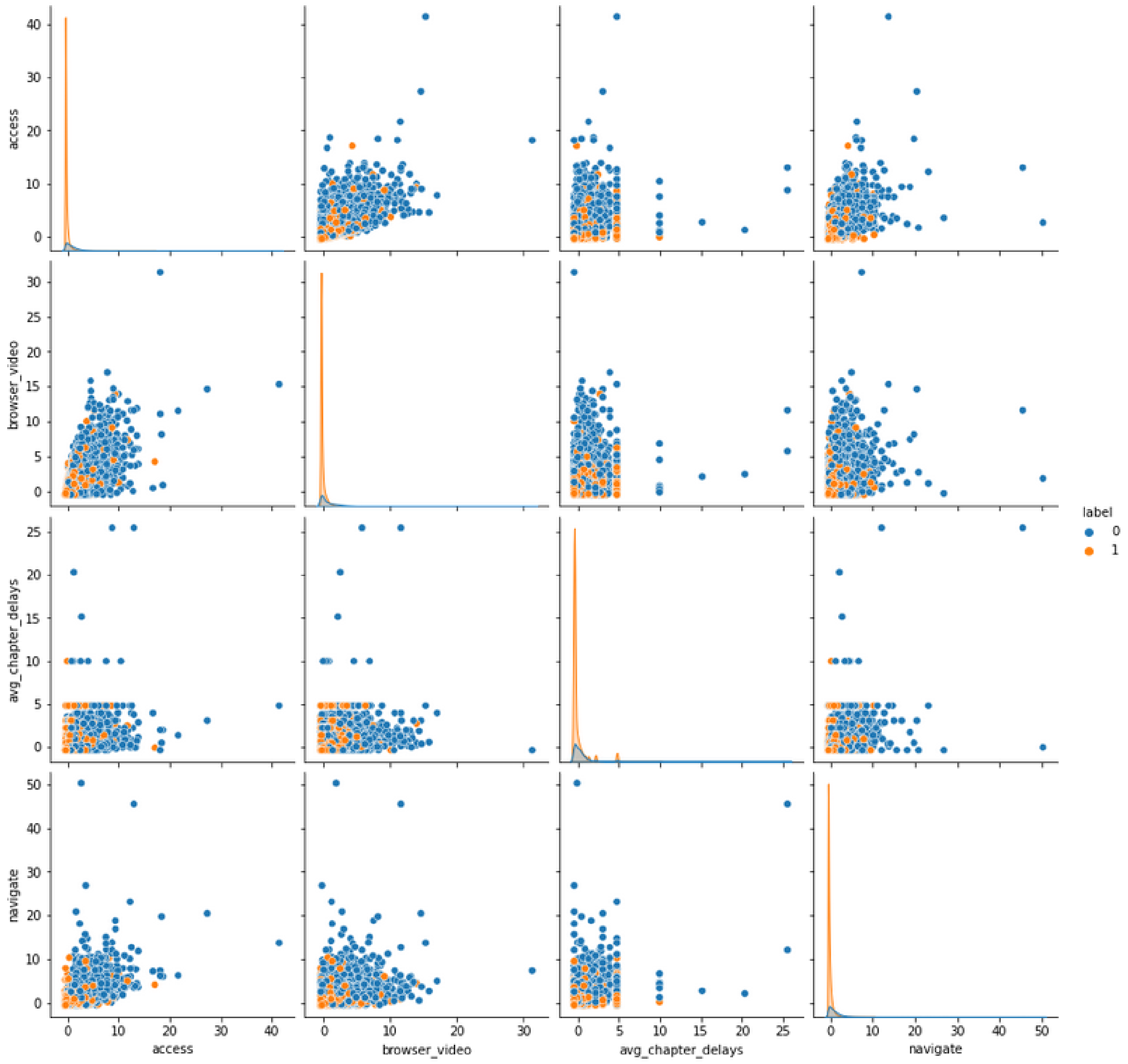


Figure 5.11: Pairplot of selected features

5.14 Comparison with previous works

Model	Recall Score
CLSA	86.50%
FWTS	86.50%
DP-CNN	87.17%
C-RF	88.90%
CNN-SE-GRU	97.26%
Our model (Stacking)	**97.64%**
CLMS-Net	98.80%

Table 5.20: Performance Comparison with previous Models

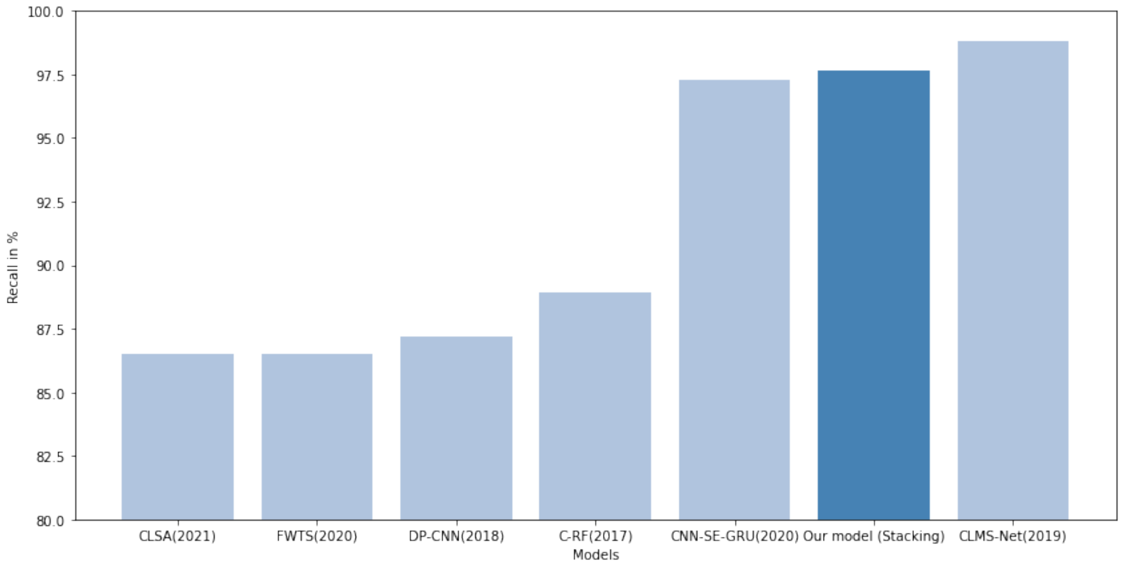


Figure 5.12: Comparison with previous Models

We compared our stacking model with previous models that were run on the KDDCUP 2015 dataset. All the previous models were implemented on the time-series dataset within KDDCUP, whereas our model is done on the non-sequential data from the same data source. Both datasets contained data from the same students, as the same enrolment IDs were present. Since we did not find any other previous work that dealt with the featureVectorWithLabel.csv file from KDDCUP 2015, we believe this is the closest form of comparison. From the figure 5.12, it is clear that our model performed better than all other models in recall score except for the CLMS-Net which is slightly better. This proves that ensemble learning, in particular stacking, outperformed traditional machine learning and custom deep learning models.

Chapter 6

Model Explanation Using XAI

6.1 LIME Interpretation

6.1.1 KNN: non-dropout test sample

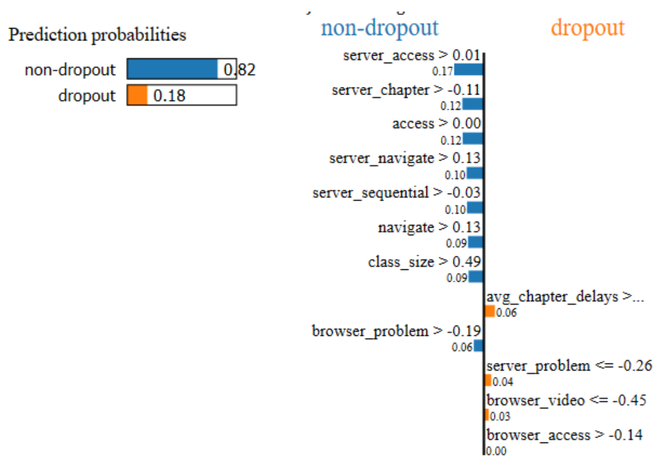


Figure 6.1: KNN: Prediction Probabilities of test sample

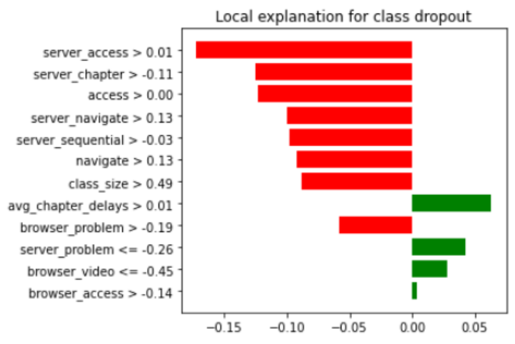
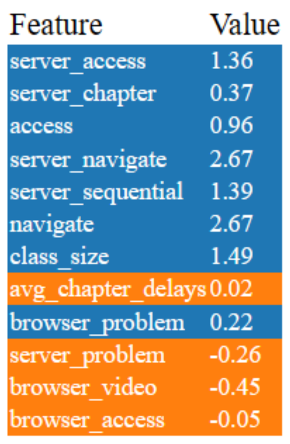


Figure 6.2: KNN: local explanation for class dropout



Feature	Value
server_access	1.36
server_chapter	0.37
access	0.96
server_navigate	2.67
server_sequential	1.39
navigate	2.67
class_size	1.49
avg_chapter_delays	0.02
browser_problem	0.22
server_problem	-0.26
browser_video	-0.45
browser_access	-0.05

Figure 6.3: KNN: Feature Values of the test sample

This specific sample is explained through LIME based on the KNN model. We can see that it has a non-dropout probability of 0.82 and a dropout probability of 0.18 in figure 6.1, meaning this data point is more likely to be predicted as a non-dropout. The features that support this decision are server-access, server-chapter, access, server-navigate, and a few others as shown by the local explanation graph 6.2. The values of those mentioned features in figure 6.3 are above a certain threshold that causes them to sway in one direction. The red features are highly correlated with the dropout label and the green features are highly correlated with the dropout label. Since the red outweighs the green, it has a higher chance of being a non-dropout prediction. In this example, server-access has a negative correlation value of 0.17 which is greater than the threshold value of 0.01 as explained by LIME.

6.1.2 GBM: non-dropout test sample

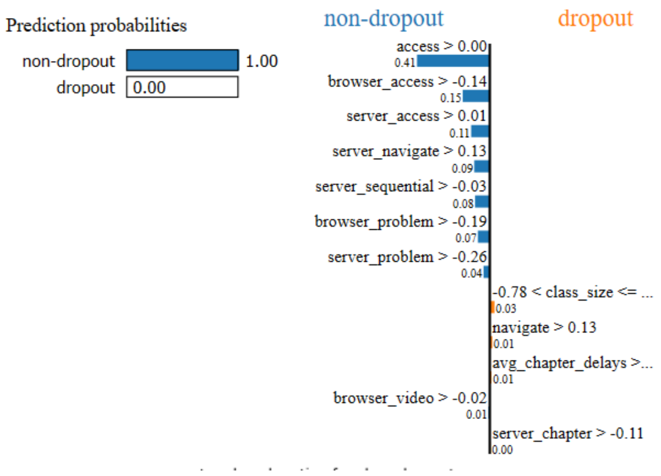


Figure 6.4: GBM: Prediction Probabilities of test sample

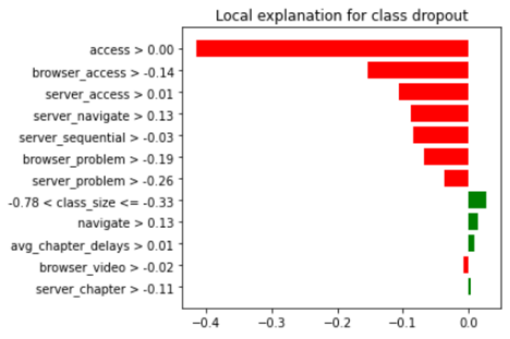
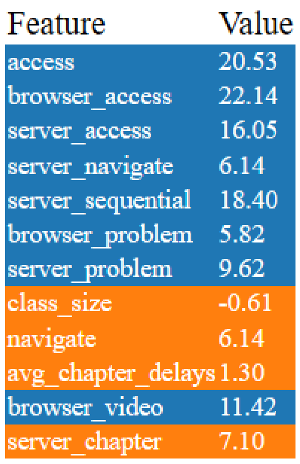


Figure 6.5: GBM: local explanation for class dropout



Feature	Value
access	20.53
browser_access	22.14
server_access	16.05
server_navigate	6.14
server_sequential	18.40
browser_problem	5.82
server_problem	9.62
class_size	-0.61
navigate	6.14
avg_chapter_delays	1.30
browser_video	11.42
server_chapter	7.10

Figure 6.6: GBM: Feature Values of the test sample

GBM model gives another explanation for the same test sample as in KNN. It predicts the sample will definitely be a non-dropout, as the probability of label 0 is

1.0 and label 1 is 0 in figure 6.4. This is due to the fact that access, browser-access, and server-access have a high positive value which indicates that a learner is actively using the MOOC platform since the feature values shown in figure 6.6 are measured as frequencies. LIME justifies this result, by choosing these features as the most contributing to a non-dropout sample.

6.1.3 RFC: Dropout test sample

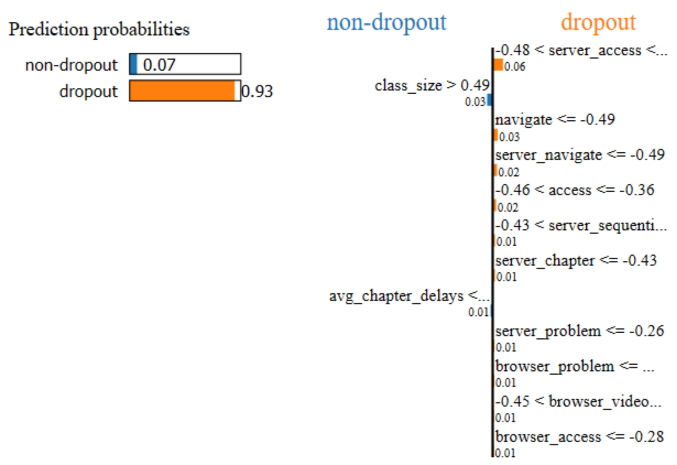


Figure 6.7: RFC: Prediction Probabilities of test sample

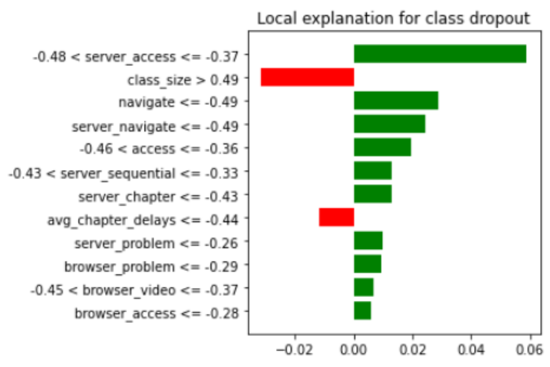
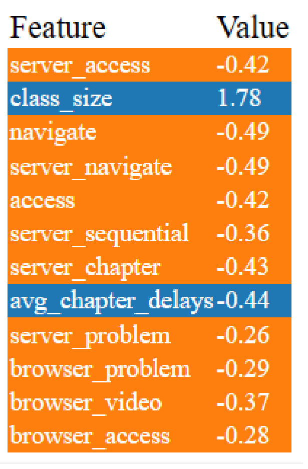


Figure 6.8: RFC: local explanation for class dropout



Feature	Value
server_access	-0.42
class_size	1.78
navigate	-0.49
server_navigate	-0.49
access	-0.42
server_sequential	-0.36
server_chapter	-0.43
avg_chapter_delays	-0.44
server_problem	-0.26
browser_problem	-0.29
browser_video	-0.37
browser_access	-0.28

Figure 6.9: RFC: Feature Values of the test sample

We used our RFC model to give an explanation for a dropout sample. LIME gives a probability of 0.93 in the dropout class and 0.07 in the non-dropout class, as highlighted by figure 6.7. The top 3 features that are behind this decision are server-access, navigate, server-navigate, and access. We can see that these features were also given the most priority during non-dropout explanation by KNN and GBM models. From the feature values in figure 6.9, we can clearly see that the orange features have negative values, indicating that this student did not interact with the MOOC platform that much, which resulted in a dropout.

6.1.4 MLP: Dropout test sample

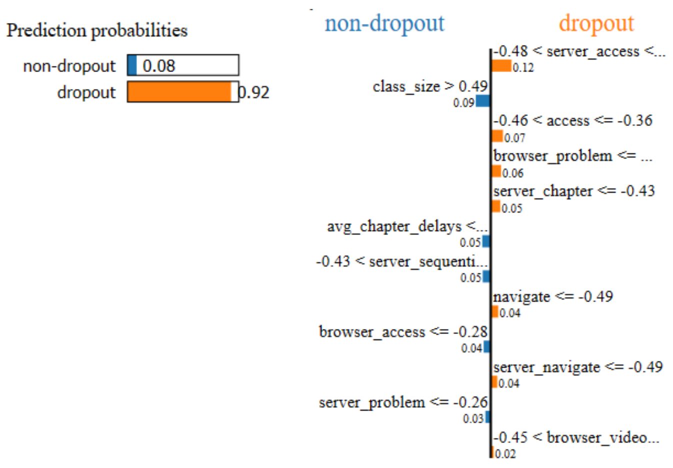


Figure 6.10: MLP: Prediction Probabilities of test sample

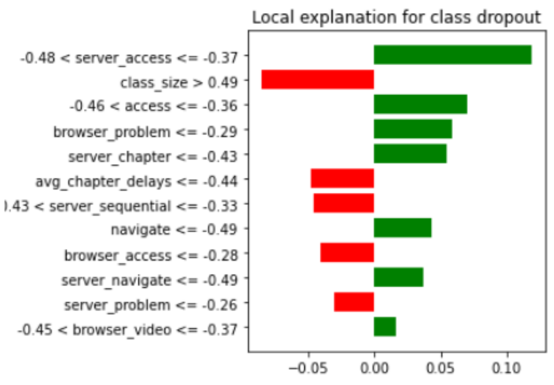
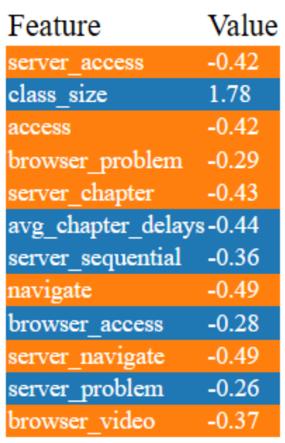


Figure 6.11: MLP: local explanation for class dropout



Feature	Value
server_access	-0.42
class_size	1.78
access	-0.42
browser_problem	-0.29
server_chapter	-0.43
avg_chapter_delays	-0.44
server_sequential	-0.36
navigate	-0.49
browser_access	-0.28
server_navigate	-0.49
server_problem	-0.26
browser_video	-0.37

Figure 6.12: MLP: Feature Values of the test sample

For the same dropout test sample, we used our MLP model to give an explanation to compare with RFC. The explanation is almost identical, LIME gives a probability of 0.92 in the dropout class and 0.08 in the non-dropout class in figure 6.10. The features that are behind this decision are server-access, access, browser-problem, and server-chapter as shown in figure 6.11. Class size has a strong correlation with the non-dropout class, which was seen from KNN's explanation. From the feature values in figure 6.12, we can see that the magnitude of the contribution of green features is much higher than the red ones, resulting in a prediction of dropout.

6.1.5 Decision Tree: Mixed probability test sample

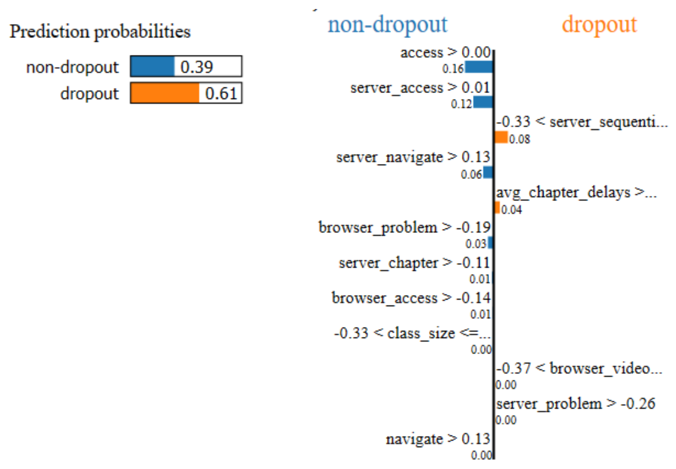


Figure 6.13: Decision Tree: Prediction Probabilities of test sample

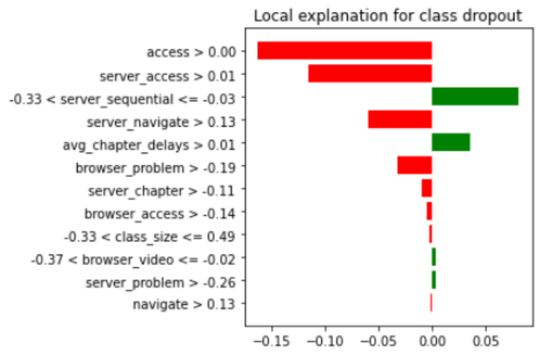
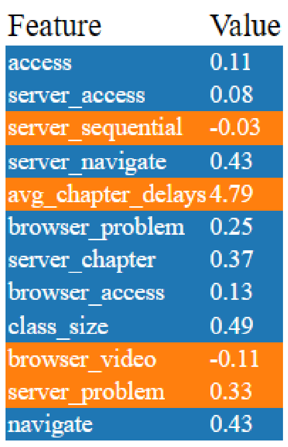


Figure 6.14: Decision Tree: local explanation for class dropout



Feature	Value
access	0.11
server_access	0.08
server_sequential	-0.03
server_navigate	0.43
avg_chapter_delays	4.79
browser_problem	0.25
server_chapter	0.37
browser_access	0.13
class_size	0.49
browser_video	-0.11
server_problem	0.33
navigate	0.43

Figure 6.15: Decision Tree: Feature Values of the test sample

We tested another sample, which gave us a less definitive prediction using the decision tree model. Here, LIME gave the non-dropout class a probability of 0.39 and the dropout class a probability of 0.61 in figure 6.13. The figure 6.14 shows access and server-access as the highest contributing factor to the non-dropout class, while server-sequential and avg-chapter delays are positively correlated to the dropout class. From our understanding, this should have been a non-dropout prediction and the feature values shown by figure 6.15 are mostly positive, which suggests that the student has used the MOOC platform. However, LIME gives a prediction in favour of the dropout class. This type of result shows the inaccuracy of our models, which is reflected in our recall and precision metric, which is not 100%.

6.1.6 LIME interpretation of other models

SVM

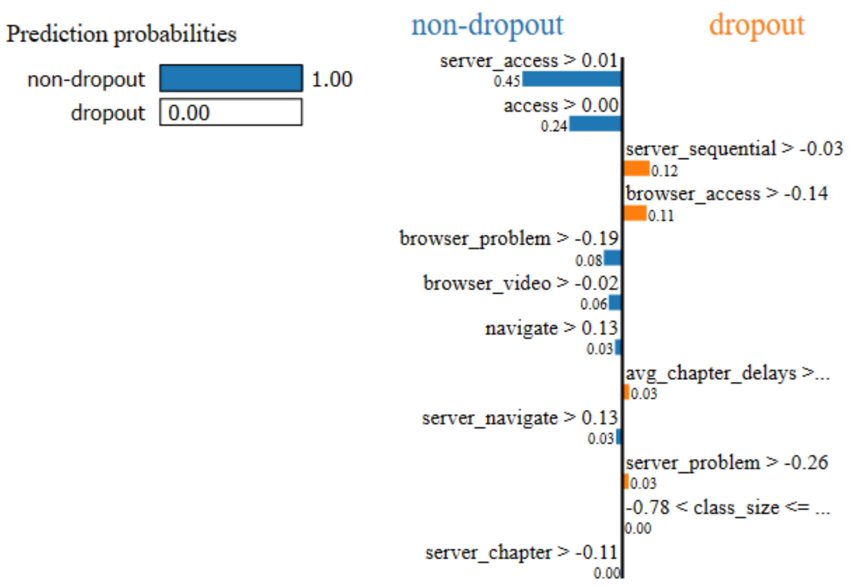


Figure 6.16: SVM: Prediction Probabilities of test sample

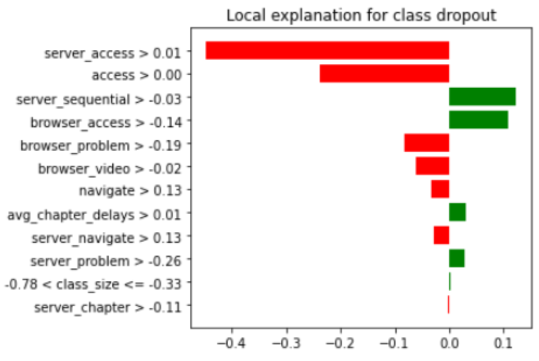
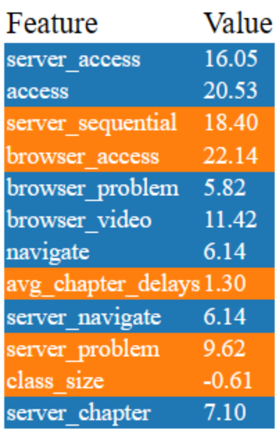


Figure 6.17: SVM: local explanation for class dropout



Feature	Value
server_access	16.05
access	20.53
server_sequential	18.40
browser_access	22.14
browser_problem	5.82
browser_video	11.42
navigate	6.14
avg_chapter_delays	1.30
server_navigate	6.14
server_problem	9.62
class_size	-0.61
server_chapter	7.10

Figure 6.18: SVM: Feature Values of the test sample

Naive Bayes

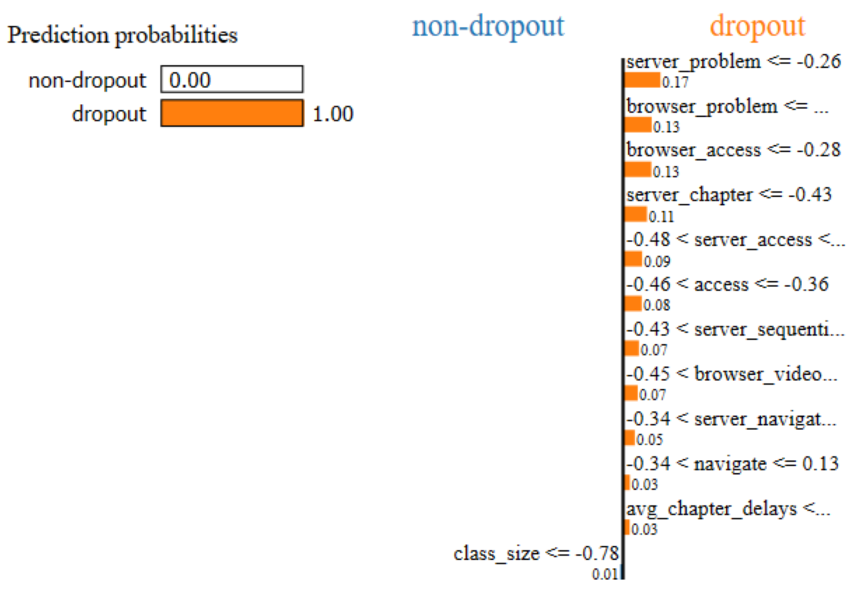


Figure 6.19: Naive Bayes: Prediction Probabilities of test sample

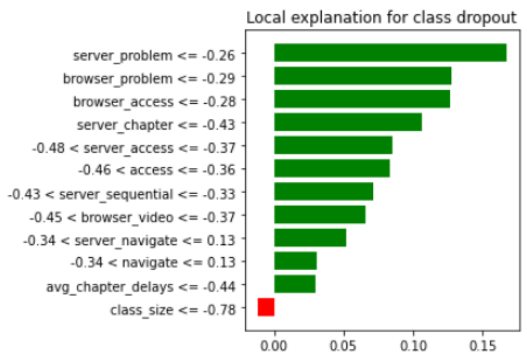
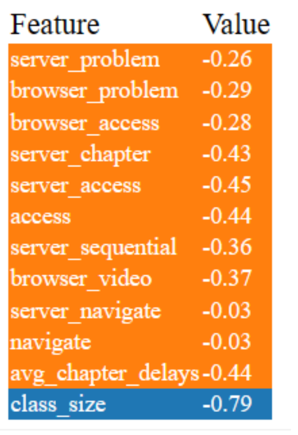


Figure 6.20: Naive Bayes: local explanation for class dropout



Feature	Value
server_problem	-0.26
browser_problem	-0.29
browser_access	-0.28
server_chapter	-0.43
server_access	-0.45
access	-0.44
server_sequential	-0.36
browser_video	-0.37
server_navigate	-0.03
navigate	-0.03
avg_chapter_delays	-0.44
class_size	-0.79

Figure 6.21: Naive Bayes: Feature Values of the test sample

Soft Voting Classifier

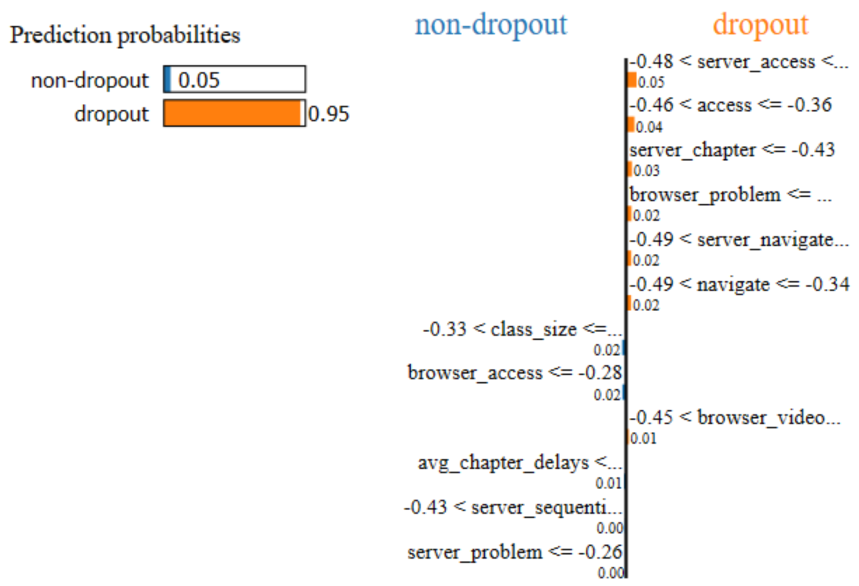


Figure 6.22: Soft Voting Prediction Probabilities of test sample

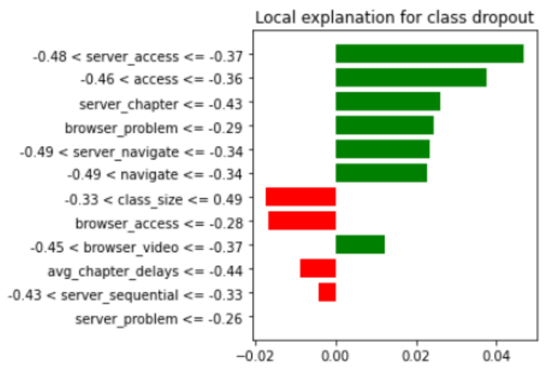
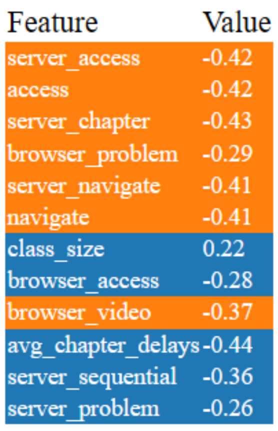


Figure 6.23: Soft Voting local explanation for class dropout



Feature	Value
server_access	-0.42
access	-0.42
server_chapter	-0.43
browser_problem	-0.29
server_navigate	-0.41
navigate	-0.41
class_size	0.22
browser_access	-0.28
browser_video	-0.37
avg_chapter_delays	-0.44
server_sequential	-0.36
server_problem	-0.26

Figure 6.24: Soft Voting Feature Values of the test sample

6.2 SHAP Interpretation

6.2.1 Soft Voting Classifier: kernel explainer

SHAP is another way to explain our model's decision-making. Unlike LIME, SHAP provides both local and global explanations. The SHAP force plot can be used to see the effect of each feature on the prediction. The base value for the Shapely value is the average of all predictions. Since we have 2 classes, the explainer will have 2 base values, one for label 0 and another for label 1.

Local explanation

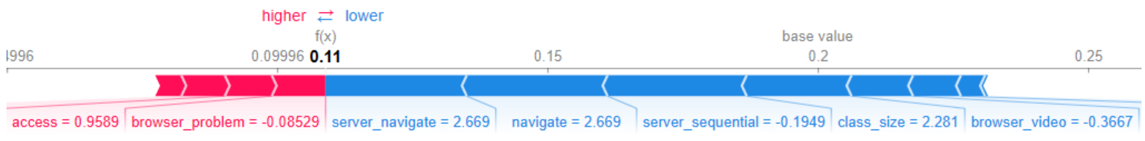


Figure 6.25: Soft Voting: With respect to the non-dropout class

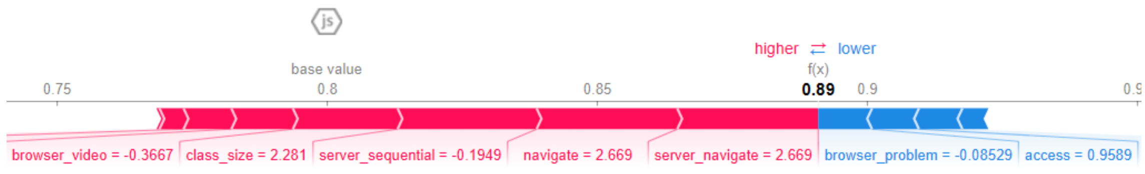


Figure 6.26: Soft Voting: With respect to the dropout class

We used our Soft Voting Classifier to explain an instance of dropout. In the first force plot in figure 6.25, the explanation of the observation is with respect to the non-dropout class. The red features are driving the base value higher, whereas the blue features are trying to bring down the base value. The model output value is 0.11 which is lower than the base value of 0.2 for non-dropout. So it has a lower probability of non-dropout. In the second plot in figure 6.26, we can see that the base value for the dropout class is 0.8, and the model output value is 0.89. This higher value suggests a greater chance of this sample being predicted as a dropout. Features such as server-navigate, navigate and server-sequential correlate to higher dropout probability, and features like browser-problem and access push the predictions toward non-dropout.

Global explanation

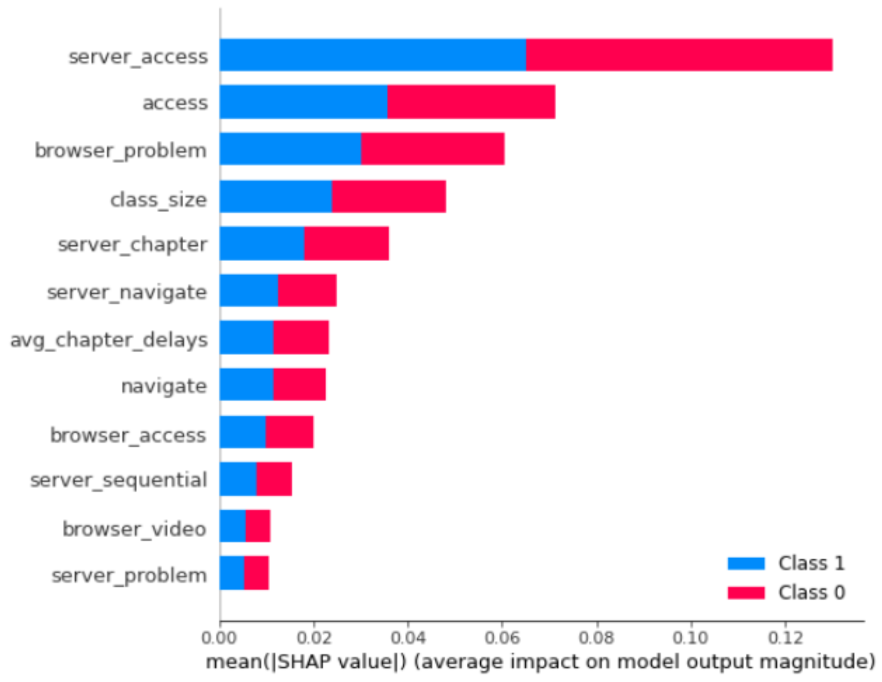


Figure 6.27: Soft Voting Summary plot

SHAP can use a subset or a sample of data points to give a global explanation of the entire model. Based on 20 samples, the mean SHAP value gives the average effect of each feature on the prediction. From the figure 6.27, we can see that server-access, access, and browser-problem are the top 3 features that contribute the most to the prediction.

6.2.2 KNN: Kernel explainer

Local explanation

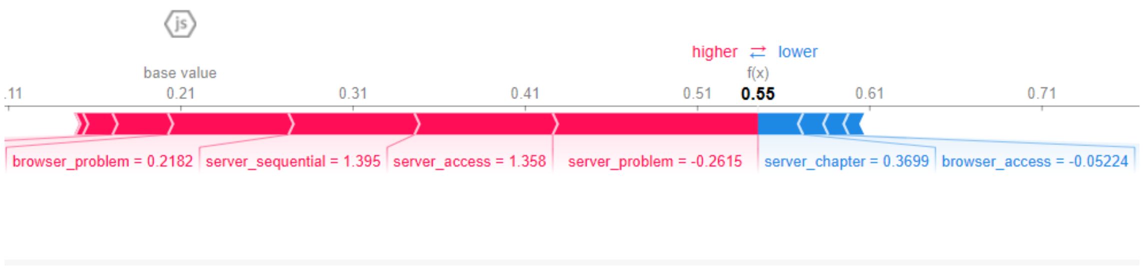


Figure 6.28: KNN: With respect to the non-dropout class

The above plots show an explanation for a non-dropout sample using the KNN model. In the first figure 6.28, the explanation of the observation is with respect to the non-dropout class. The red features are driving the base value higher, whereas the blue features are trying to bring down the base value. The model output value is 0.55 which is higher than the base value of 0.21 for non-dropout. So it has a

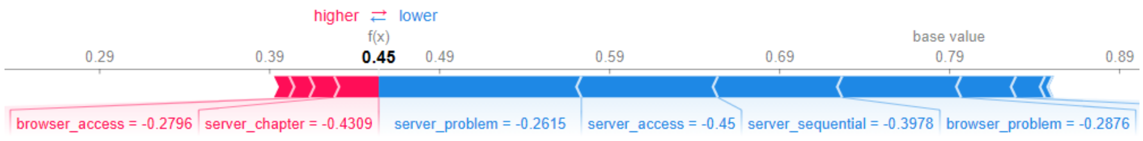


Figure 6.29: KNN: With respect to the dropout class

higher probability of non-dropout. In the second figure 6.29, we can see that the base value for the dropout class is 0.79, and the model output value is 0.45. A lower value suggests a lesser chance of this sample being predicted as a dropout, and instead predicting it as a non-dropout. Features such as server-problem, server-access, and server-sequential correlate to higher non-dropout probability, and features like browser-access and server-chapter push the predictions toward the dropout class.

Global explanation

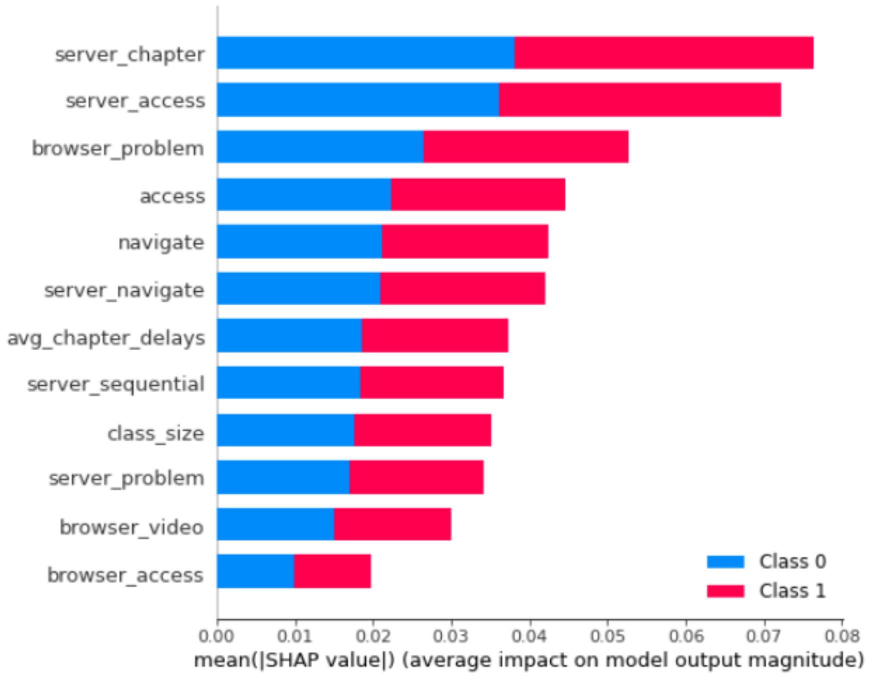


Figure 6.30: KNN Summary plot

As for the global explanation, we used 20 samples to approximate the results. Compared with our voting classifier model, features like server-access and browser problem are common as the top features contributing to our prediction, as shown in figure 6.30. KNN choose server-chapter as the most important feature, which is ranked lower in other models' explanations. But browser-video and server-problem are 2 features that are ranked lower in both models.

6.2.3 Decision Tree: Tree explainer

Local explanation

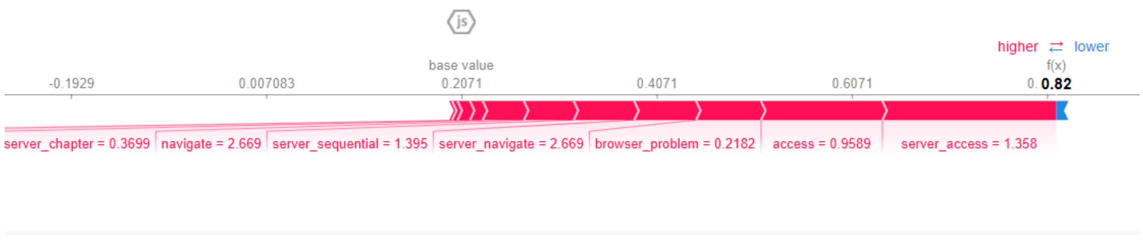


Figure 6.31: Decision Tree: With respect to the non-dropout class

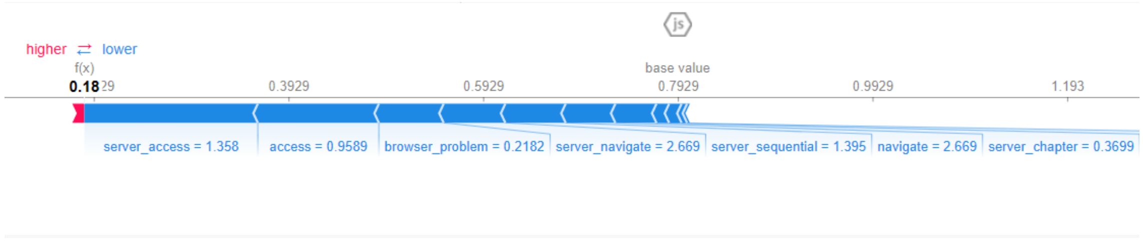


Figure 6.32: Decision Tree: With respect to the dropout class

Here is another sample which has a very high probability of being a non-dropout. The force plot in figure 6.31 shows server-access, access, and browser-problem having the highest driving force in predicting non-dropout with a model output of 0.82 which is above the base value of 0.2071. With respect to the dropout class in figure 6.32, we can see those same features have the driving effect of lowering the probability of a dropout with a low model output of 0.18 which is much lower than the base value of 0.7929 of the dropout class.

6.2.4 Global explanation of other models

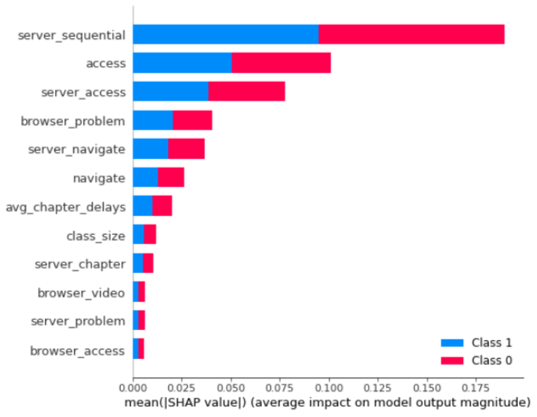


Figure 6.33: Decision Tree summary plot

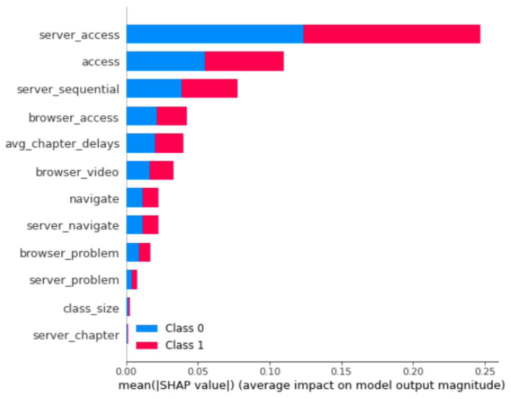


Figure 6.34: SVC summary plot

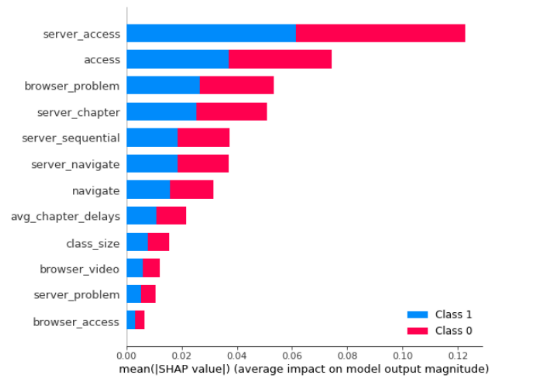


Figure 6.35: RFC summary plot

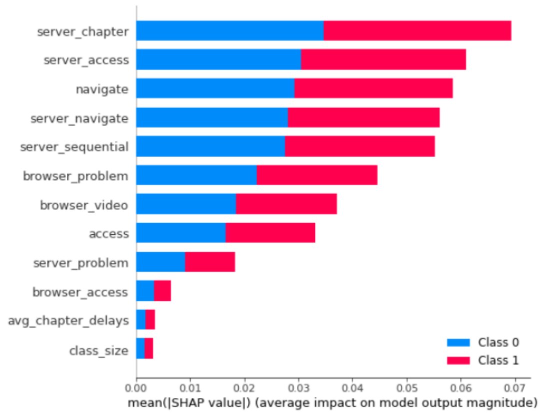


Figure 6.36: Naive Bayes summary plot

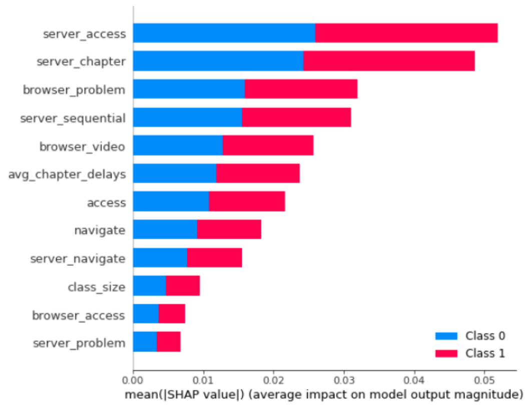


Figure 6.37: Stacking summary plot

From the figures 6.33, 6.34, 6.35,6.36, and 6.37 we can conclude that features like access, server-access, and browser-problem are contributing the most to the prediction of both classes. Similarly, features like browser-video, server-problem and class-size are impacting the decision the least.

6.2.5 MLP: Correlation between features and classes

The figure 6.38 shows high server-access value, which means it has a strong negative contribution to the prediction. In our case, it refers to the non-dropout class. High values of Access and server-sequential have a positive correlation with the dropout prediction class. Additionally, we can see that class size, avg-chapter-delays has very little contribution to our prediction.

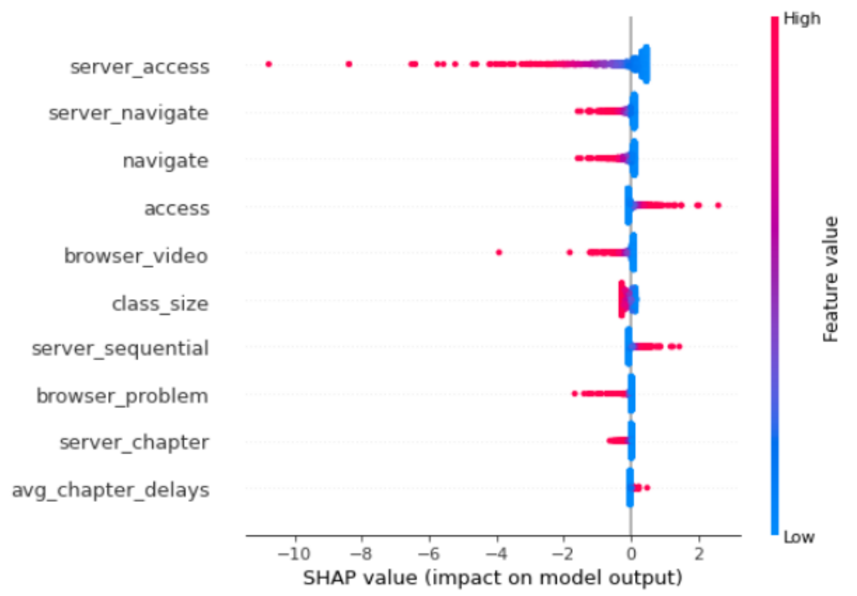


Figure 6.38: MLP Summary dot plot

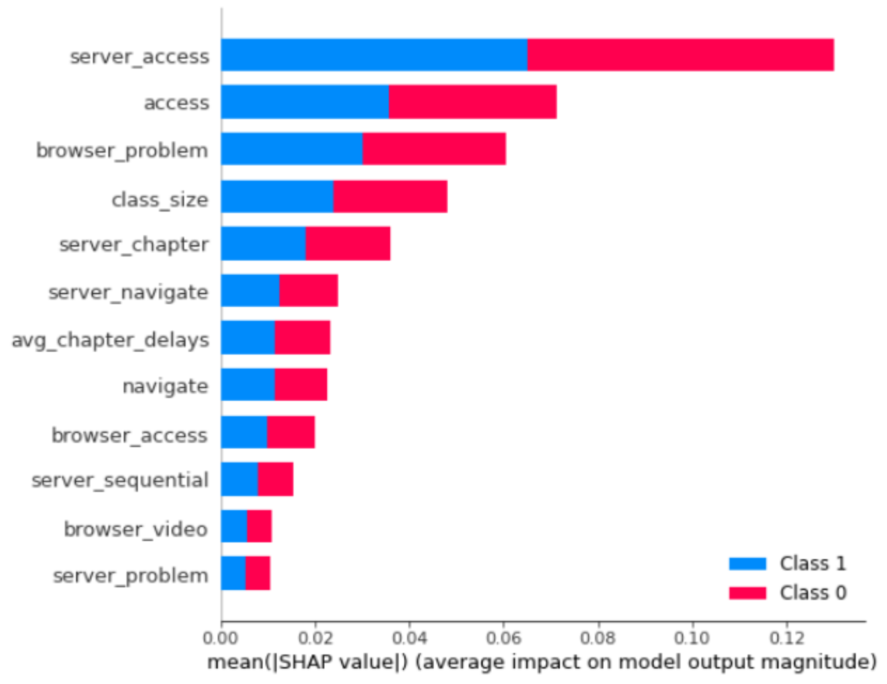


Figure 6.39: MLP Summary bar plot

Chapter 7

Conclusion and Future Work

7.1 Conclusion

MOOCs were thought to bring a new era of education that redefines the way we learn. To create a vast platform which is open and accessible to anyone sounds promising for people of every background. However, the extremely high rate of dropout overshadows the benefits that MOOCs come with and is hindering its growth. To prevent early withdrawal behaviour of students and encouraging them to continue is a vital task that researchers are exploring. Although various models have been developed to solve this challenge, our research uses different ensemble learning techniques to approach this problem. This method has already been proven to outperform other baseline machine learning models in other fields of research and competitions [4]. Therefore, this research attempts to find a solid solution that enables MOOCs to avoid significant student dropout rates. From our experimental results, we have shown this method to be effective with a recall score of 97.636%. Furthermore, we have shown that our models were successful in reducing variance compared to baseline models. Finally, we tried to explain our models using XAI and find the most important features that effect dropouts.

7.2 Future Work

Our first initiative in data preprocessing and result analysis exhibited promising scores from our models. However, we can conduct further studies on our existing models in order to receive more optimal results. Since we have applied Explainable AI (XAI) to comprehend the decisions made by our models, we know which features contribute the most to dropout. We could make an AI chatbot to respond to inactive users who have not accessed the MOOC platforms and did not interact with its content that much. The dataset we worked on is based on China's MOOC platform, so we could explore other areas of the world to find out how students of different culture behave in regard to dropout. Similarly, a potential future work could be to study the relationship between various course subjects and dropout to see whether any specific subject have a correlation. Another difficult area of MOOC dropout problem is whether a learner is completely dropping out and not pursuing further studies, or are they switching to a different platform due to not liking some aspects of the previous MOOC platform.

Bibliography

[1] P. H. Swain and H. Hauska, "The decision tree classifier: Design and potential," *IEEE Transactions on Geoscience Electronics*, vol. 15, no. 3, pp. 142–147, 1977. DOI: 10.1109/TGE.1977.6498972.

[2] T. Evgeniou and M. Pontil, "Support vector machines: Theory and applications," vol. 2049, Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[3] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, pp. 367–378, 2002.

[4] G. Valentini and F. Masulli, "Ensembles of learning machines," vol. 2486, May 2002, pp. 3–22, ISBN: 978-3-540-44265-3. DOI: 10.1007/3-540-45808-5_1.

[5] P. Cunningham and S. Delany, "K-nearest neighbour classifiers," *Mult Classif Syst*, vol. 54, Apr. 2007. DOI: 10.1145/3459665.

[6] M.-C. Popescu, V. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, Jul. 2009.

[7] A. Cutler, D. Cutler, and J. Stevens, "Random forests," *Machine Learning - ML*, vol. 45, pp. 157–176, Jan. 2011. DOI: 10.1007/978-1-4419-9326-7_5.

[8] Y. Lee and J. Choi, "A review of online course dropout research: Implications for practice and future research," *Educational Technology Research and Development*, vol. 59, pp. 593–618, Oct. 2011. DOI: 10.1007/s11423-010-9177-y.

[9] Vikramkumar, V. B, and Trilochan, *Bayes and naive bayes classifier*, 2014. DOI: 10.48550/ARXIV.1404.0933. [Online]. Available: https://arxiv.org/abs/1404.0933.

[10] M. H. Baturay, "An overview of the world of moocs," *Procedia - Social and Behavioral Sciences*, vol. 174, pp. 427–433, Feb. 2015. DOI: 10.1016/j.sbspro.2015.01.685.

[11] K. Jordan, "Massive open online course completion rates revisited: Assessment, length and attrition," Jun. 2015. DOI: 10.13140/RG.2.1.2119.6963.

[12] J. Whitehill, J. Williams, G. Lopez, C. Coleman, and J. Reich, "Beyond prediction: First steps toward automatic intervention in mooc student stopout," Jun. 2015.

[13] M. Josek, J. Fernandez, A. Perez-Encinas, B. Zimonjic, L. De Vocht, and M. Falisse, "The international-friendliness of universities: Research report of the esnsurvey 2016," *Erasmus Student Network AISBL, Brussels, Belgium*, 2016.

[14] M. Ribeiro, S. Singh, and C. Guestrin, *"why should I trust you?": Explaining the predictions of any classifier*, San Diego, California, Jun. 2016. DOI: 10.18653/v1/N16-3020. [Online]. Available: https://aclanthology.org/N16-3020.

[15] B. Hong, Z. Wei, and Y. Yang, "Discovering learning behavior patterns to predict dropout in mooc," *2017 12th International Conference on Computer Science and Education (ICCSE)*, pp. 700–704, 2017.

[16] S. Lundberg and S.-I. Lee, *A unified approach to interpreting model predictions*, Dec. 2017.

[17] T. Miller, *Explanation in artificial intelligence: Insights from the social sciences*, 2017. DOI: 10.48550/ARXIV.1706.07269. [Online]. Available: https://arxiv.org/abs/1706.07269.

[18] R. S. Baragash and H. Al-Samarraie, "An empirical study of the impact of multiple modes of delivery on student learning in a blended course," *The Reference Librarian*, vol. 59, no. 3, pp. 149–162, 2018. DOI: 10.1080/02763877.2018.1467295. eprint: https://doi.org/10.1080/02763877.2018.1467295. [Online]. Available: https://doi.org/10.1080/02763877.2018.1467295.

[19] R. Baragash and H. Al-Samarraie, "Blended learning: Investigating the influence of engagement in multiple learning delivery modes on students' performance," English, *Telematics and Informatics*, vol. 35, no. 7, pp. 2082–2098, Oct. 2018, ISSN: 0736-5853. DOI: 10.1016/j.tele.2018.07.010.

[20] F. Dalipi, A. S. Imran, and Z. Kastrati, "Mooc dropout prediction using machine learning techniques: Review and research challenges," in *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2018, pp. 1007–1014. DOI: 10.1109/EDUCON.2018.8363340.

[21] P. Hall, *On the art and science of machine learning explanations*, 2018. DOI: 10.48550/ARXIV.1810.02909. [Online]. Available: https://arxiv.org/abs/1810.02909.

[22] J. Lee, A. Hong, and J. Hwang, "A review of massive open online courses: Mooc's approach to bridge the digital divide," Jun. 2018.

[23] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, *Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai*, 2019. DOI: 10.48550/ARXIV.1910.10045. [Online]. Available: https://arxiv.org/abs/1910.10045.

[24] D. Sun, Y. Mao, J. Du, P. Xu, Q. Zheng, and H. Sun, "Deep learning for dropout prediction in moocs," in *2019 Eighth International Conference on Educational Innovation through Technology (EITT)*, 2019, pp. 87–90. DOI: 10.1109/EITT.2019.00025.

[25] H. Aldowah, H. Al-Samarraie, A. I. Alzahrani, and N. Alalwan, "Factors affecting student dropout in moocs: A cause and effect decision-making model," *Journal of Computing in Higher Education*, vol. 32, pp. 429–454, 2020.

[26] D. Bashir, G. D. Montañez, S. Sehra, P. S. Segura, and J. Lauw, *An information-theoretic perspective on overfitting and underfitting*, 2020. arXiv: 2010.06076. [Online]. Available: https://arxiv.org/abs/2010.06076.

[27] M. Dudjak and G. Martinovic, "In-depth performance analysis of smote-based oversampling algorithms in binary classification," vol. 11, p. 2020, Apr. 2020. DOI: 10.32985/ijeces.11.1.2.

[28] R. Kumar, *Adaboost and gradient boost – comparitive study between 2 popular ensemble model techniques*, Oct. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/10/adaboost-and-gradient-boost-comparitive-study-between-2-popular-ensemble-model-techniques/.

[29] S. C. Pulikottil and M. Gupta, "Onet – a temporal meta embedding network for mooc dropout prediction," *2020 IEEE International Conference on Big Data (Big Data)*, pp. 5209–5217, 2020.

[30] S. Yin, L. Lei, H. Wang, and W. Chen, "Power of attention in mooc dropout prediction," *IEEE Access*, vol. 8, pp. 202 993–203 002, Jan. 2020. DOI: 10.1109/ACCESS.2020.3035687.

[31] Y. Zheng, Z. Gao, Y. Wang, and Q. Fu, "Mooc dropout prediction using fwts-cnn model based on fused feature weighting and time series," *IEEE Access*, vol. 8, pp. 225 324–225 335, 2020. DOI: 10.1109/ACCESS.2020.3045157.

[32] S. G. An, *Introduction to how an multilayer perceptron works but without complicated math*, Oct. 2021. [Online]. Available: https://medium.com/codex/introduction-to-how-an-multilayer-perceptron-works-but-without-complicated-math-a423979897ac.

[33] Q. Fu, Z. Gao, J. Zhou, and Y. Zheng, "Clsa: A novel deep learning model for mooc dropout prediction," *Comput. Electr. Eng.*, vol. 94, no. C, Sep. 2021, ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2021.107315. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2021.107315.

[34] T. Jo, "Ensemble learning," pp. 285–307, Jan. 2021. DOI: 10.1007/978-3-030-65900-4_13.

[35] E. Magalhaes, G. Santos, F. Molina, J. P. Javidi da Costa, F. Mendonca, and R. de Sousa Junior, "Student dropout prediction in mooc using machine learning algorithms," Nov. 2021, pp. 1–6. DOI: 10.1109/WCNPS53648.2021.9626227.

[36] K. Mrhar, M. Abik, and O. Douimi, "A dropout predictor system in moocs based on neural networks," *Journal of Automation Mobile Robotics Intelligent Systems*, vol. 14, p. 72, Mar. 2021. DOI: 10.14313/JAMRIS/4-2020/48.

[37] W. contributors, *Random forest*, Dec. 2022. [Online]. Available: https://en.wikipedia.org/wiki/Random_forest.

[38] E. Owens, B. Sheehan, M. Mullins, M. Cunneen, J. Ressel, and G. Castignani, *Explainable artificial intelligence (xai) in insurance*, 2022.

[39] IBM, *What is a decision tree?* IBM, Ed. [Online]. Available: https://www.ibm.com/topics/decision-trees.

[40] IBM, *What is the k-nearest neighbors algorithm?* IBM, Ed. [Online]. Available: https://www.ibm.com/topics/knn.

[41] Javatpoint, *Support vector machine algorithm*, Javatpoint, Ed. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm.