Supervised Machine Learning: Predicting User Video Engagement with Educational Videos

By Alex Holme

ABSTRACT

The following study evaluates the effectiveness of various machine learning algorithms in predicting user engagement with educational videos. Utilizing the "VLEngagement" dataset, which includes viewing habits from a popular OER repository, models were trained using the SciKitLearn library. The algorithms tested included Logistic Regression, Random Forest, Gradient Boost Machines (GBM), Support Vector Machines (SVM), and K Nearest Neighbor (KNN). Models were evaluated using a five fold ROC-AUC mean score. The Random Forest model achieved the highest score of 0.888, closely followed by GBM with 0.883, both outperforming other models with a collective mean score of 0.829. This superior performance may be attributed to their ability to handle complexity and non-linearity effectively.

After hyperparameter tuning, the Random Forest model's ROC-AUC score improved marginally to 0.891. The model demonstrated a high weighted average F-1 score of 0.93, excelling in classifying non-engaging videos (F-1 score of 0.96) but underperforming with engaging videos (F-1 score of 0.56, Recall of 0.47). The imbalance in the dataset led to a bias towards non-engaging video predictions, resulting in frequent false negatives for engaging videos.

1. INTRODUCTION

Usage of Massive Open Online Courses (MOOCs) like Coursera has grown rapidly. The outbreak of the Covid-19 pandemic accelerated this growth further. Between 2019 and 2021 unique users registered on Coursera more than doubled, rising from 44 million to 92 million (https://www.weforum.org/agenda/2022/01/online-learning-courses-reskill-skills-gap).

In response to the increase in demand for online learning, online course programs have been rapidly expanding their catalogue. In 2023 alone, Coursera added over 1,200 new courses (https://blog.coursera.org/what-the-world-learned-on-coursera-in-2023-and-next-years-must-know-skills/).

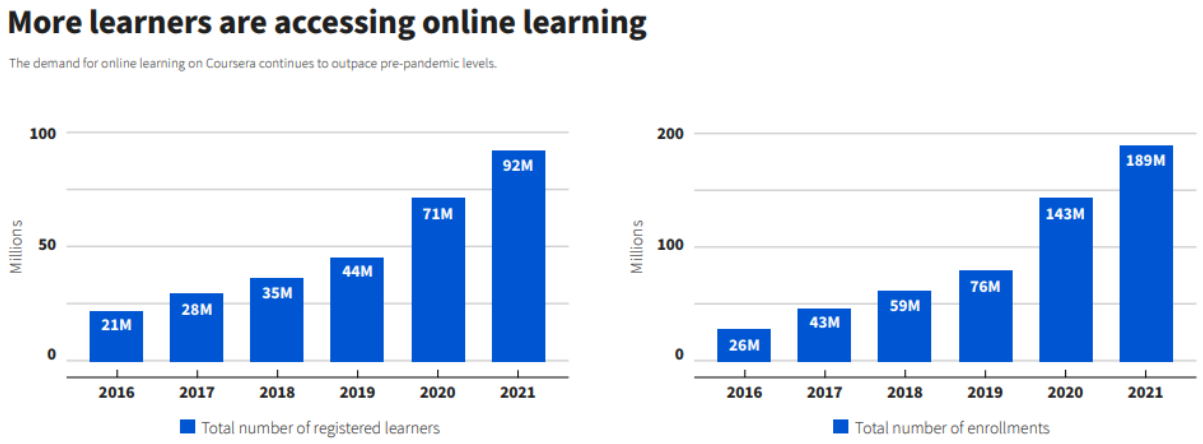

Figure 1: Growth of online learning. Published in the 2021 Coursera Impact Report

This has increased the challenge of matching viewers with appropriate videos that will capture and hold their interest. One simple technique platforms can start with when deciding how to promote a video is to first determine if the video is generally engaging. The following report seeks to do just that using a VLE Dataset put together by researcher Sahan Bulathwela at University College London. The dataset includes features of many educational videos as well as a target label if a video was engaging or not. A video is considered engaging when the median percentage of the video watched across all viewers was at least 30%.

This report will compare several different machine learning algorithms to see how effective they are at predicting classifications. This report seeks to evaluate which models would be most appropriate at future video engagement prediction tasks.

2. **DATASET**

The data used in this report was captured as part of a 2020 study "VLEngagement: A Dataset of Scientific Video Lectures for Evaluating Population-based Engagement". The dataset was made from "aggregated video lectures consumption data coming from a popular scientific OER repository, VideoLectures.Net". Data was taken from lecturers that were conducted between September 1, 1999 and October 1, 2017. Viewer data was logged between December 8, 2016 and February 17, 2018.

2.1 **Features**

The following features are included in the dataset:

title_word_count - the number of words in the title of the video.

document_entropy - a score indicating how varied the topics are covered in the video, based on the transcript. Videos with smaller entropy scores will tend to be more cohesive and more focused on a single topic.

freshness - The number of days elapsed between 01/01/1970 and the lecture published date. Videos that are more recent will have higher freshness values.

easiness - A text difficulty measure applied to the transcript. A lower score indicates more complex language used by the presenter.

fraction_stopword_presence - A stopword is a very common word like 'the' or 'and'. This feature computes the fraction of all words that are stopwords in the video lecture transcript.

speaker_speed - The average speaking rate in words per minute of the presenter in the video.

silent_period_rate - The fraction of time in the lecture video that is silence (no speaking).

Target Variable:

engagement - Target label for training. True if learners watched a substantial portion of the video (30%), or False otherwise.

2.2 **Data Processing**

The dataset used was processed prior to posting the CSV file online. No further processing was required. The dataset includes 9239 videos.

3. **MODELS USED**

Five of the most common machine learning models were selected to be compared against one another. These included: Logistic Regression, Random Forest, Gradient Boosting, Support Vector Machine, and K Nearest Neighbour. Each of these models will be compared using default parameters against each other, with a "best model" being selected. The best model will then undergo hyperparameter tuning to optimize it, and a final prediction will be made.

Model coefficients are optimized using an 80:20 test split. 80% of the data is used for training, with the target known to the model. 20% of the data is classified as test data, where the target value is obscured from the model. Models optimize their coefficients by optimizing their training and test prediction accuracy. Test data is used to help prevent overfitting, to prevent the model from memorizing how to achieve perfect accuracy for the training data, in a way that it will not be able to handle the introduction of new data.

3.1 **Logistic Regression**

Broadly speaking, logistic regression algorithms take many input features and compute a binary output of either true or false, indicating its target evaluation of true or false.

Logistic Regression first computes a linear combination of the features. This is the sum of a bias term and a weighted sum of the features. Part of training a logistic regression model is optimizing the coefficients of each feature to produce the most accurate results. For the linear combination to function properly, each feature must be weighted equally. The built in SciKitLearn function "StandardScaler()" was used which places all features on a scale of 0-1.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

Equation 1: Linear Combination Equation

With the linear combination calculated, a sigmoid function is then used, which calculates the probability between 0 and 1, corresponding to the likelihood of the input belonging to the positive class. Scikit Learn uses a default threshold of 0.5. If the probability is above 0.5 it assigns that videos target prediction as engaging, if it is below 0.5 it will assign it as not engaging.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Equation 2: Sigmoid Function

A cost function is added to encourage simplified models. Models that heavily weight all features, thus increasing model complexity, are penalized. This helps prevent overfitting of the data. The weight of this penalty is imposed by the coefficient C. The effect of C on the strength of the penalty is inverse, a larger C will impose a smaller penalty on complexity. Different cost equations can be used including L1 (Lasso Regression), or L2 (Ridge Regression).

During hyperparameter tuning, C values of 0.001, 0.01, 0.1, 1, 10, and 100 were tested. L1 (Lasso) and L2 (Ridge) were tested. Liblinear and Saga classifier solvers were both tested.

3.2. **Random Forest**

Random Forests are a form of enable algorithms. They take multiple decision tree models and use them together where the total effectiveness is better than any individual model. To help optimize model effectiveness and prevent overfitting each decision tree is constructed differently, and will be exposed to a different random split of data.

Once all decision trees are created, when the model is exposed to new data, each individual tree predicts a classification. The random forest model then uses a majority voting scheme to make a single classicisation based on the classifications of each tree.

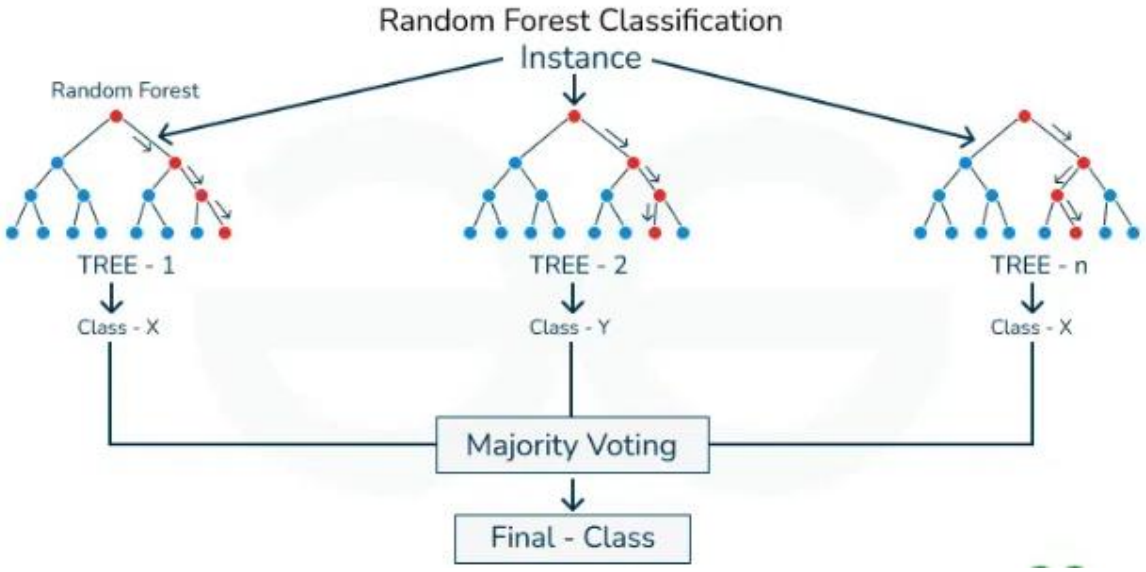

Figure 2: Random Forests Visualized

Within SciKitLearn users can alter several random forest parameters including the number of trees used (the default value is 100), minimum number of samples required to split an internal node (the default value is 2), and the maximum depth of trees (by default no maximum depth is enforced).

During hyperparameter tuning, values of 100, 200 and 300 were tested for the number of trees. Values of 2, 5, and 10 were tested for the minimum number of sample splits. Values of None, 10, 20, and 30 were tested for the maximum depth.

3.3 **Gradient Boosting**

An ensemble algorithm that constructs a series of trees. The addition of each tree seeks to minimize the total error. Gradient boosting trees tend to use large amounts of shallow trees.

Within SciKitLearn users can alter the number of trees used by the model (n_estimators) with a default value of 100 trees, the learning rate (learning_rate) which controls the impact of each tree to the final model and has a default value of 0.1, and the maximum depth of trees (max_depth) which has a default depth of 3.

During hyperparameter tuning, values of 100, 200 and 300 for the number of trees was tested. A learning rate of 0.01, 0.1, and 0.3 were tested. A maximum depth of 3, 4, and 5 were tested.

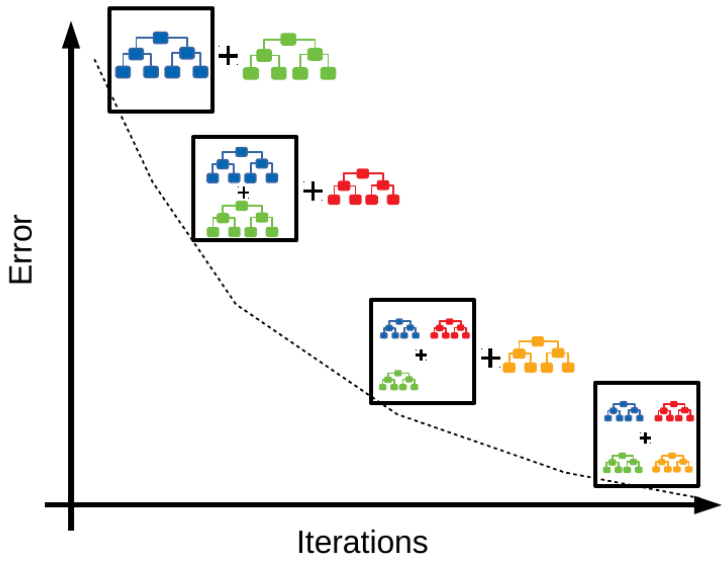

Figure 3: Simplified Visualization of Gradient Boosting Algorithms

3.4 **Support Vector Machine**

Support Vector Machines seek to find a line or hyperplane, depending on dimensionality, that best separates the data into different classes. SVMs will maximize the margin between the line or hyperplane and the nearest data points. The nearest data points are referred to as Support Vectors. Like previous models, SVMs require features to be scaled consistently, and the default SciKitLearn StandardScaler() function was applied.

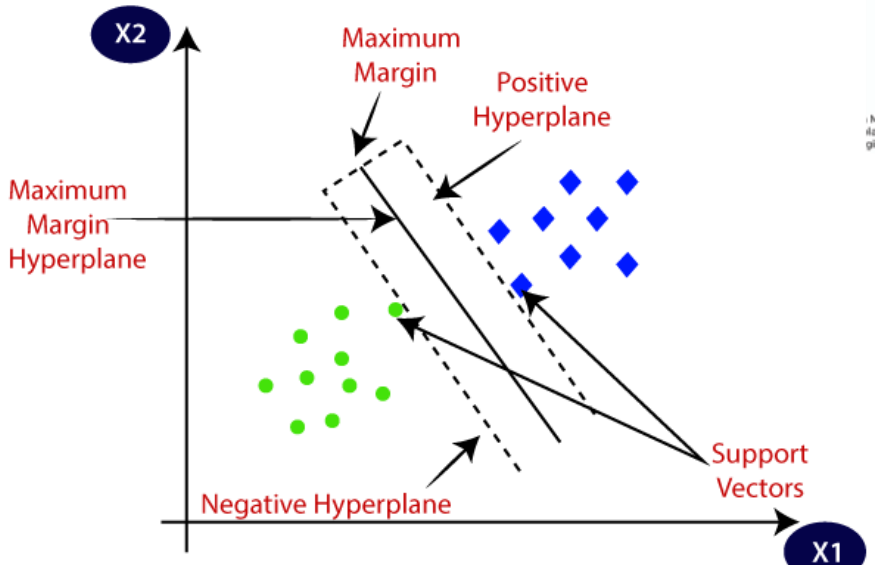

Figure 4: 2-Dimensional Support Vector Machine

When data exists in clusters it can be difficult for a linear line or hyperplane to separate clusters. For this, a kernelized support vector helps. This is the process of taking data into a higher

dimension, where it's location in this new dimension is a function of its location in the original dimension. This results in data clustered together to being placed in close proximity in this new

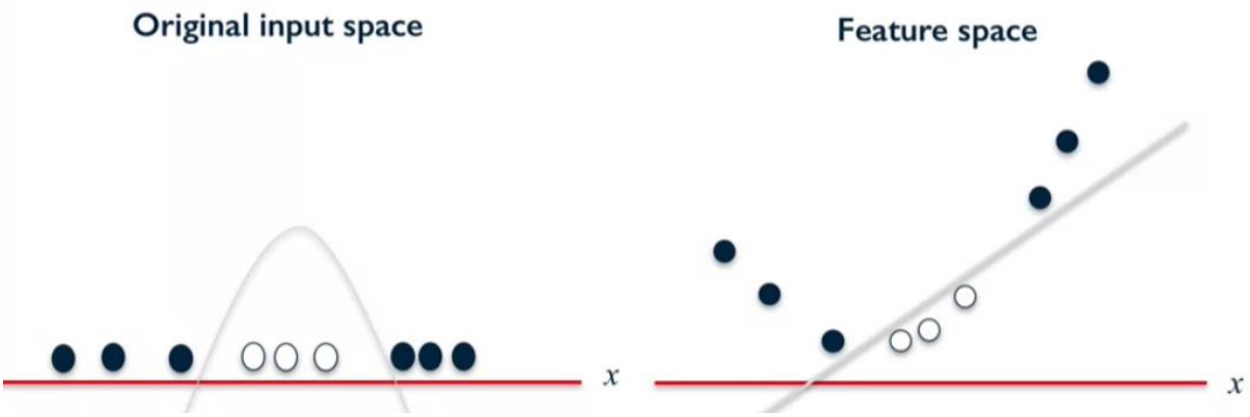

Figure 5: Kernelized Support Vector of 1 feature data.

dimension. Data can then be easily separated by a linear function.

For the purpose of this report, Linear and Radial Basis Function Kernels were tested. Linear kernels compute the dot product of features. This works well with data with large number of features relative to the number of samples, and is generally computationally efficient.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

Equation 3: Linear Kernel Dot Product Equation

Radial Basis Function Kernels (RBF) is a function that creates a Gaussian curve in a higher dimension than the data. RBF contain a gamma term that influences the tightness of the fit. With a larger gamma there is a tighter fit of clustered data, a smaller gamma leads to a smoother fit.

$$K(x, x') = \exp\left(-\gamma \parallel x - x' \parallel^2\right)$$

Equation 4: RBF Equation with a gamma term

During hyperparameter tuning, a linear and RBF kernel were tested. Values of gamma tested include 'scale', 'auto' (see equation below), 0.1, and 0.01, and a classifier C of 0.1, 1, and 10 (complexity penalty).

$$Scale: \frac{1}{n_{\text{features}} \cdot \text{Var}(X)} \quad Auto: \frac{1}{n_{\text{features}}}$$

Equation 5: SciKitLearn Scale and Auto gamma values

3.5 **K Nearest Neighbour**

K Nearest Neighbour (K-NN) is a classification algorithm that assigns a target class to an object based on its closest neighbour(s) in a training set. A single or multiple neighbours can be considered. If multiple neighbours are used, all neighbours can be weighted equally with a simple majority vote to identify the target class, or closer neighbours can have a more heavily weighted vote. Like previous models, K-NN requires features to be scaled consistently, and the default SciKitLearn StandardScaler() function was applied.

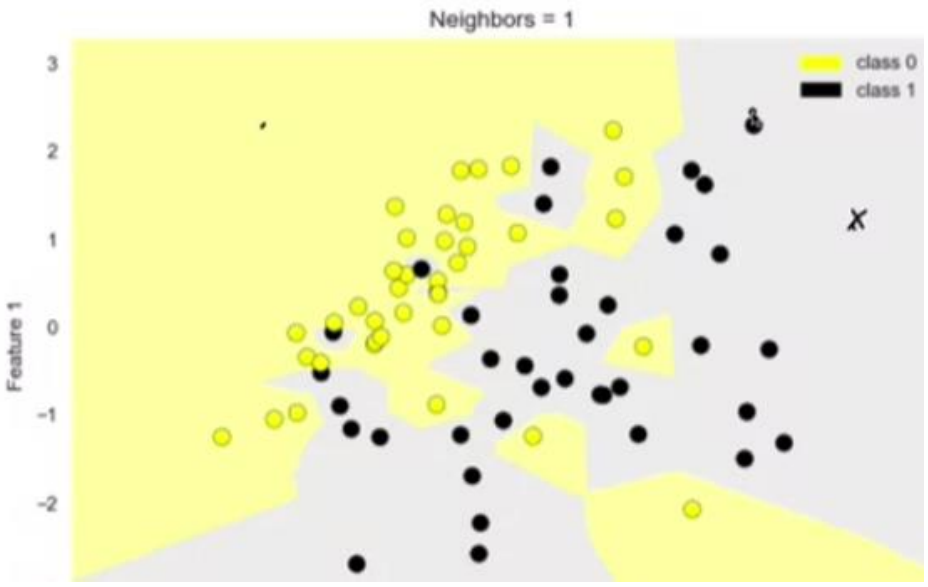

Figure 6: A plot of target class decision areas based on a 1 neighbour approach

Within SciKitLearn users can define the number of neighbours to be considered in the target class vote, with a default value of 5. Users can select the how the voting scheme works, with a default uniform weight set where all neighbours have equal weight regardless of who is closest. And users can select how the distance between points is calculated, with the default being Euclidean distance.

During hyperparameter tuning, 3, 5, 7, 9, and 11 were tested for the number of neighbours. A uniform classifier weight was tested as well as a distance based weighting scheme where closer training points were weighted more heavily. Distances were calculated using Euclidean and Manhattan distance (see equations below).

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Equation 6: Euclidean Distance Equation

$$d(x,y) = | x_1 - y_1 | + | x_2 - y_2 | + \cdots + | x_n - y_n |$$

Equation 7: Manhattan Distance Equation

3.6 **Model Evaluation**

All five models were tested for their prediction making abilities against the test data using their default SciKitLearn parameters. Models were evaluated based on their mean ROC-AUC values (Receive Operating Characteristic, and Area Under the Curve). ROC curves are a plot of the True Positive Rate (Or the number of true positive predictions divided by the sum of the true positives and false negative predictions), and the False Positive Rate (False positives divided by the sum of False positives and true negatives). The area under this curve is calculated, a higher AUC indicates higher rates of correct predictions with low numbers of false negatives. ROC-AUC is a standard evaluation metric of supervised machine learning algorithms.

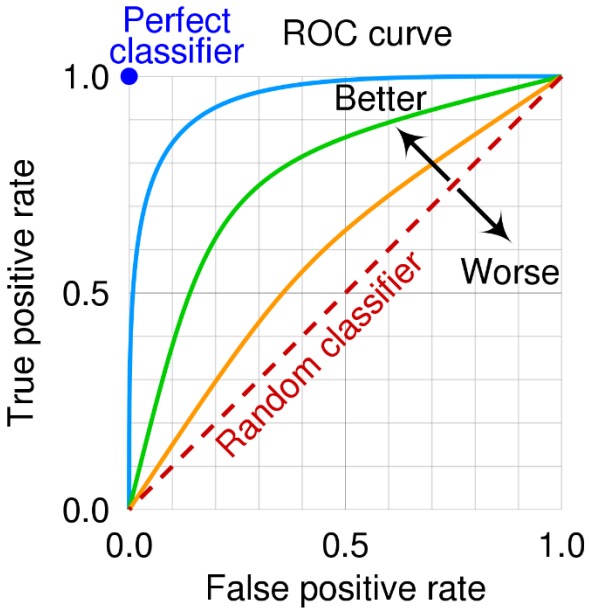

Figure 7: ROC Curve Plot

For further accuracy, a cross validation technique was used that split the data five times in different ways using the SciKitLearn cross_val_score() function. The mean of those 5 splits were taken and used to compare models, hence "mean ROC-AUC".

3.7 **Hyperparameter Tuning**

Once the best performing model was selected, a grid search hyperparameter tuning function was applied. This allows users to input a grid of multiple values for multiple parameters. Each unique combination of parameters in the grid is tested, with each combination producing a mean ROC-AUC score with the data being split five times. The combination of parameters that obtain the highest ROC-AUC score are selected for any future classification task.

3.8 **Evaluation Metrics**

Additional metrics were used to evaluate the final tuned model to evaluate its performance. These included Accuracy, Recall (True Positive Rate), Precision (True positives divided by the sum of true positives and false positives), and an F1 score (combing precision and recall into a single metric). A high recall score indicates the model will not often make false negative predictions. A high Precision score indicates the model will not often make false positive scores. There are certain applications where the weight of one metric might be higher than the other. For example, with credit card fraud, a false positive case of fraud may cause a user a bit of headache to have to contact customer support, but this is generally better than a fraudster escaping detection. A false negative is worse than a false positive, so a fraud detection model should optimize recall score even at the expense of precision.

For video engagement, a false positive and a false negative were thought to generally have equal weight, thus overall accuracy and F1 were treated as important.

Classes often will not contain the same number of objects. Classes can either be macro averaged, where the prediction of each class individually is weighted the same, regardless of size. Or classes can be weighted averaged where larger classes are weighted more heavily. For this dataset, many more videos were found to be not engaging or False than engaging or positive. Both weighted and macro averaged results will be presented.

3.9 **Feature Ranking**

Random Forrest and Gradient Boosting Machines can calculate how important each feature is. Random Forrests calculates how much each feature reduces the impurity in the tree. GBMs similarly calculate the contribution of each feature to the reduction in loss across all trees in the ensemble.

Understanding the importance of each feature can be highly valuable to the content platform as well as its creators. Knowing which specific aspects of videos drive up engagement can help creators optimize their videos for engagement and drive up how long users stay engaged with the platform at large.

Feature ranking can also help identify data leakage. Data leakage occurs when training data contains information about the target variable, that would not be present to a model in a real-world scenario. If a feature has a much higher importance than other features in a non-intuitive way, it is worth investigating if there is some error in that feature that allows the model to "cheat" its way to a correct prediction.

4. **RESULTS**

After running all five previously described models on the dataset, it was found that the Random Forest method was the most effective at modelling and predicting the data with a mean ROC-AUC score of 0.888 (see below). This was narrowly higher than the Gradient Boosting Machine (GBM) score of 0.883. Given the very small difference in their score relative to the standard deviation of their scores, either model could be considered appropriate for future predictions. However, the rest of this report will focus on the Random Forest Model due to its slightly higher performance.

Model	ROC-AUC Score
Logistic Regression	0.845 (+/- 0.029)
Random Forest	0.888 (+/- 0.023)
GBM	0.883 (+/- 0.021)
SVM	0.838 (+/- 0.028)
KNN	0.803 (+/- 0.033)

Figure 8: Mean ROC-AUC scores

Random Forest and GBM models are generally better at handling more complex patterns in data and non-linearity without requiring as much tunning. This may help to explain why they would perform better with default parameters selected.

Hyperparameter tunning revealed the optimal parameters given the test grid provided, were 200 trees, a minimum sample split of 5, and a maximum depth of 10. These parameters improved the mean ROC-AUC score to 0.891. This was a relatively small improvement, which helps to

validate the default values configured in SciKitLearn. Tuning the parameters only yielded a small improvement. A visualization of the first 2 splits of the final parameterized Random Forest can be seen below. The image was generated using the SKLearn plot_tree library function.

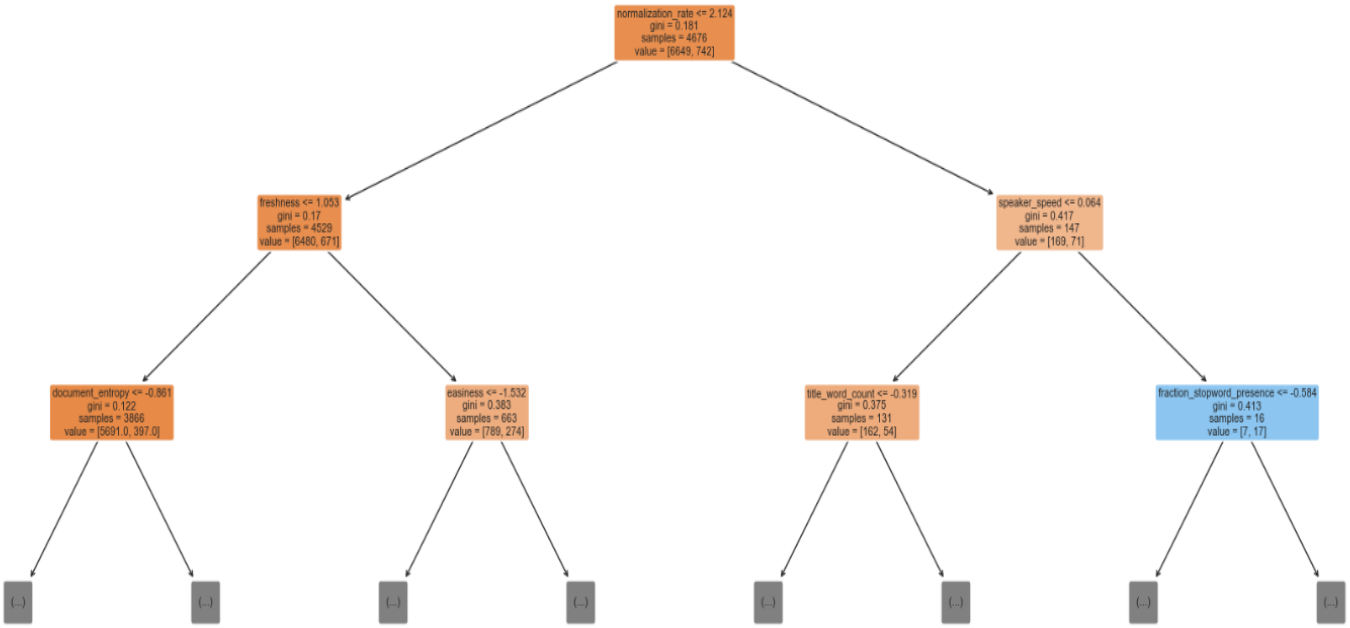

Figure 9: First 2 splits of the final parametrized random forest.

The first split occurs based on whether the normalization rate is greater than or equal to 2.124. The next 2 splits occur based on freshness and speaker speed.

With optimal parameters selected, precision, recall, and F-1 scores were calculated (see figure below). The model is considerably better at predicting non-engaging videos than engaging videos. A Recall score of 0.47 for engaging videos indicates the model was frequently making false negative predictions when presented engaging test data. The model was still able to achieve relatively high overall accuracy scores due to much larger amounts of not engaging videos being included in the data set. For further applications, stakeholders should be consulted which metrics are most important. If being able to predict if a newly uploaded video is going to be broadly engaging, more effective models could likely be trained.

	Precision	Recall	F-1	Size
Not Engaging	0.95	0.98	0.96	1683
Engaging	0.68	0.47	0.56	165
Macro Average	0.82	0.73	0.76	1848
Weighted Average	0.93	0.93	0.93	1848

Figure 10: Final Parameterized Random Forest Evaluation Metric Scores

To further understand the model, a distribution of the confidence score was plotted. Each prediction is assigned a score of 0-1 based on how confident the model is in its prediction. A large

concentration of data was located between 0.9-1.0, indicating a large amount of target labels were assigned with a high probability of being correct. Based on the Scores of the not engaging vs. engaging videos, it is likely that most of these high confidence predictions are for not engaging videos.

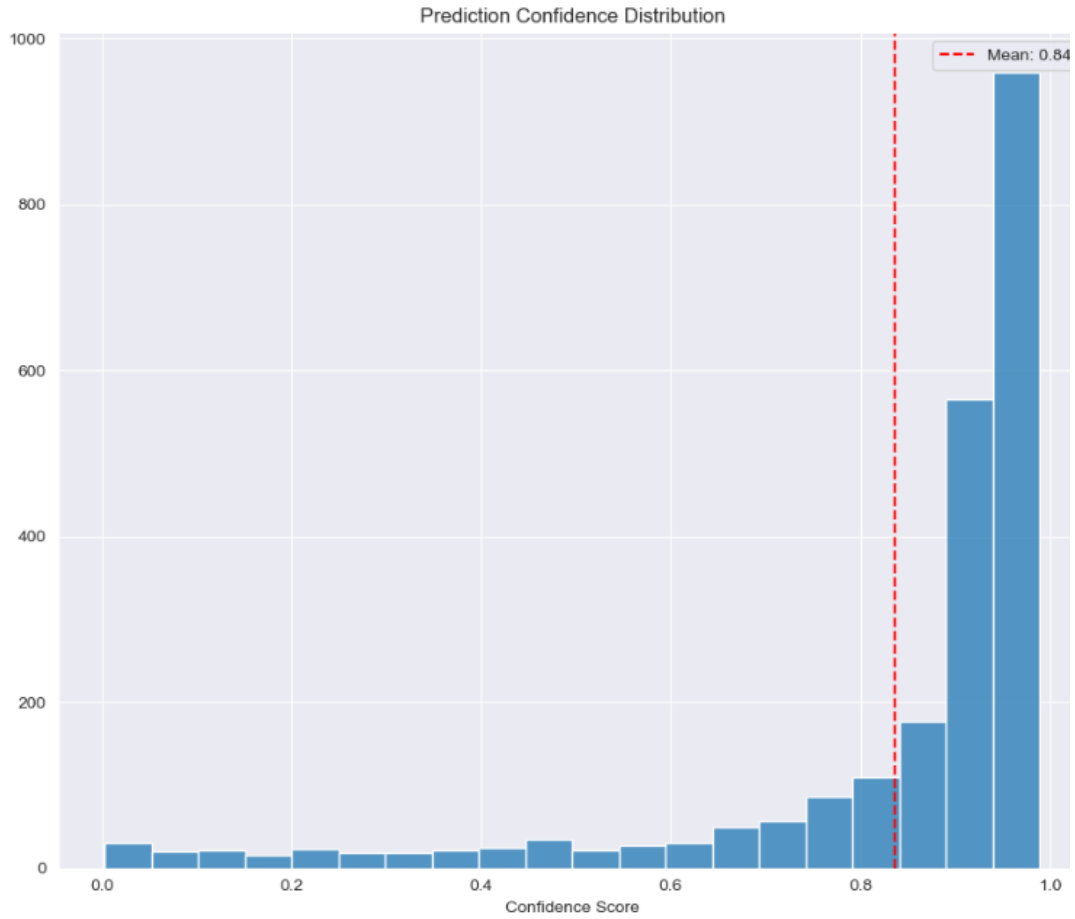

Figure 11: Prediction Confidence Distribution Plot of Finalized Model

Since a Random Forest model was selected, feature importance could be calculated. It was found the feature that by far impacted engagement the most was document_entropy (see figure below). This was a score assigned based on how much topic variety there was in a video. Further investigation is required to understand this phenomenon. How exactly do changes in topic density effect engagement? How can a creator better optimize this to drive up their engagement and viewership?

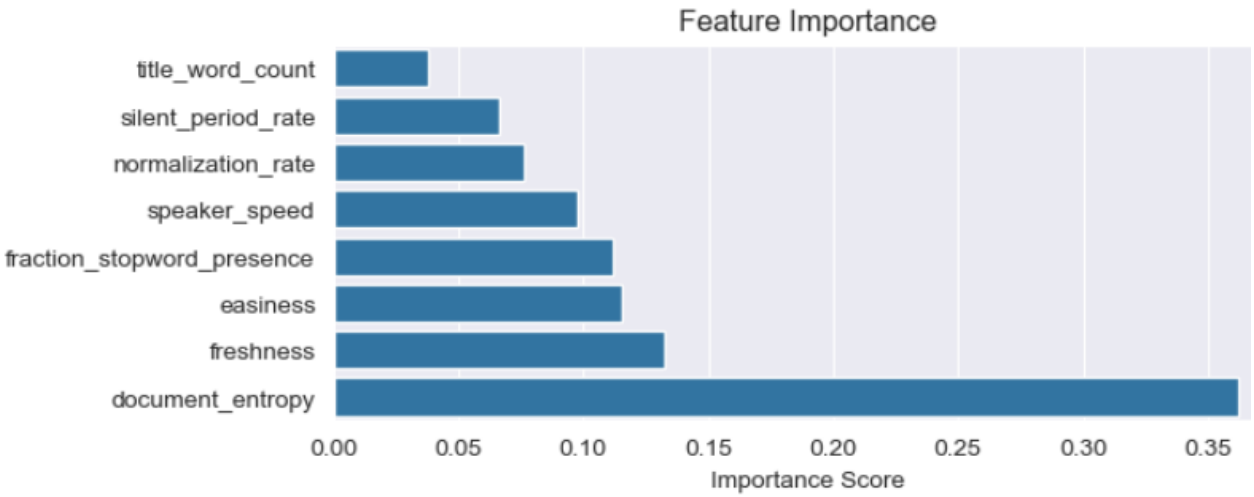

Figure 12: Calculated Feature Importance by the Finalized Model

5. **CONCLUSION**

In this study, multiple machine learning algorithms were compared on their ability to predict user engagement with educational videos. The data used was gathered as part of the study "VLEngagement: A Dataset of Scientific Video Lectures for Evaluating Population-based Engagement". The data concerns the viewing habbits of users consuming scientific lectures from a popular OER repository. The Python Library SciKitLearn was used to train various models on training and test data of educational videos. The machine learning techniques tested were Logistic Regression, Random Forest, Gradient Boost Machines (GBM), Support Vector Machines (SVM) and K Nearest Neighbour (KNN).

It was found that a random forest model generated the best overall accuracy as measured by mean ROC-AUC scores, with a score of 0.888. This was very close to the GBM score of 0.883. Both methods performed noticeably better than the other approaches which collectively had a mean score of 0.829. This may be due to Random Forest and GBM's better ability to handle complexity and non-linearity without much tuning. However, further investigation would be required to have more confidence in why these methods achieved superior performance.

A Random Forest model was selected for hyperparameter tuning, after tuning the models ROC-AUC score was improved to 0.891. This relatively small improvement helped validate the default SciKitLearn parameters, as very little improvement was yielded by optimizing parameters.

The model was able to achieve a high weighted average F-1 score of 0.93. However, this was primarily due to its strong ability to correctly classify Not Engaging videos, achieving an F-1 score of 0.96. Its performance suffered while predicting Engaging videos, achieving an F-1 score of 0.56 and a Recall of 0.47. Because there were so many more not engaging videos in the data set, models seeking overall accuracy scores were biased towards stronger performances in predicting not engaging videos than engaging videos. When presented with engaging videos, this model will produce many false negatives.

Prior to a model being applied in the real world in a production environment, stakeholders should be consulted to determine specific objects of machine learning models. This identified the best model based on overall ROC-AUC score. Since the data comprised of more non-engaging videos, this biased the model to be better at predicting non-engaging videos. This may or may not

be desirable to stakeholders. The currently configured optimal model will rarely identify a false positive engaging video. This may be desirable to a platform that wishes to have high confidence that a newly uploaded video predicted to be engaging and selected automatically to be promoted to its home page, will not disappoint and receive low engagement.