Drowsiness Detection with Machine Learning

RAVI SONI
ECE Department
STEVENS
HOBOKEN, NJ
rsoni4@stevens.edu

ABHINAV KALLAM
MECHANICAL Department
STEVENS
HOBOKEN, NJ
akallam@stevens.edu

MOHITH KUMAR GOTTIMUKULA
ECE Department
STEVENS
HOBOKEN, NJ
mgottimu@stevens.edu

***Abstract*—The challenge is to create a detection system that recognizes key characteristics of drowsiness and sends an alert when someone becomes drowsy before it is too late.**

I. Introduction

Driving while drowsy increases the likelihood of a car accident significantly. Microsleeps occur when a person falls asleep for only a few seconds5, and when they occur while driving, the car is at risk of running off the road or colliding with another vehicle. When these collisions occur at high speeds, the damage is magnified.

Drowsy driving is a major cause of automobile accidents. According to the National Highway Traffic Safety Administration (NHTSA), drowsy driving was responsible for at least 91,000 crashes in 2017, resulting in approximately 50,000 injuries and 800 deaths. According to other studies, drowsy driving causes up to 6,000 fatal crashes each year and drowsy driving is involved in approximately 21 percent of fatal car accidents [10].

II. Related Work

Han Wei et al.[14]. discuss the importance of extracting eye features for drowsiness detection and also describes extracting eye features from low resolution or degraded images.

Chirra et al.[13] use a viola-jones face detection algorithm to detect the face images and it is given as input to viola-jones eye detection algorithm.

III. Our Solution

Our approach to this problem is first to know dataset, get insights about it and find correlated variables to make use of them in training a model. We planned to test multiple machine learning algorithms, compare and conclude the solution.

A. Data souce and preprocessing

We used the Real-Life drowsiness Dataset generated by a research team from the University of Texas at Arlington specifically for detecting multi-stage drowsiness as our training and test data. The ultimate goal is for our system to be able to detect not only extreme and apparent cases of drowsiness, but also milder signals of drowsiness. Around 30 hours of films from 60 different participants make up the dataset.

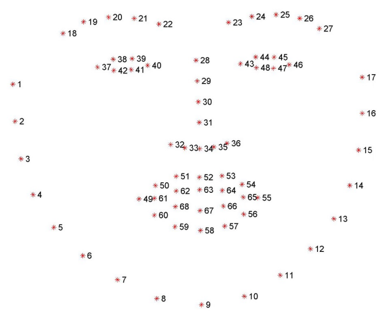


Fig. 1. Facial landmarks on Opencv

We were able to extract facial landmarks from 44 videos from 22 persons using the dataset. This allowed us to collect enough information for both the alert and drowsy states. From the 3-minute point until the end of the film, we utilized OpenCV to extract 1 frame each second.We retrieved around 240 frames each movie, totaling 10560 frames for the entire dataset, because each film was about 10 minutes long. We kept the landmarks for the eyes and mouth solely (Points 37–68) because there were 68 total markers per frame. These were the key data points from which we derived the characteristics for our model.[11]

B. Preprocessing and feature extraction

As previously mentioned, we developed relevant features for our classification model based on the face landmarks that we collected from the frames of the films. Eye aspect ratio, mouth aspect ratio, pupil circularity, and lastly, mouth aspect ratio above eye aspect ratio were the four fundamental properties that we decided on for our final models after hypothesizing and testing various features.

C. Feature Extraction

Eye Aspect Ratio

The EAR is the ratio of the length of the eyes to the width of the eyes, as the name suggests. As seen in the diagram below, the length of the eyes is measured by averaging two separate vertical lines across the eyes. Our idea was that when someone is sleepy, their eyes will shrink and they will blink more. Based on this hypothesis, we expected our model would classify a person as drowsy if their eye aspect ratio began to drop over time, i.e. their eyes became more closed or their blinking rate increased.[8]

!ht

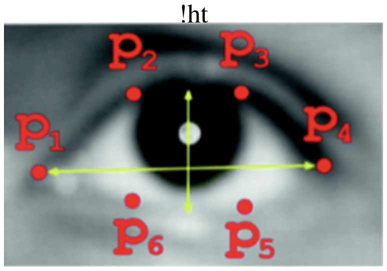


Fig. 2. Eye Aspect Ratio

!ht

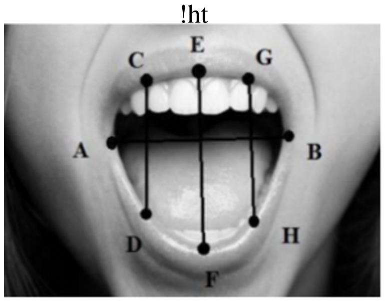


Fig. 3. Mouth Aspect Ratio

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_5||} \quad (1)$$

Mouth Aspect Ratio

The MAR, which is computationally comparable to the EAR, measures the ratio of the length of the mouth to the width of the mouth. Our theory was that as a person feels tired, they are more likely to yawn and lose control over their mouth, resulting in a larger MAR than usual.[8]

$$MAR = \frac{||EF||}{||AB||} \quad (2)$$

Pupil Circularity

PUC is a similar test to EAR, except it focuses on the pupil rather than the entire eye.[8]

$$Circularity = \frac{4 * pi * Area}{perimeter^2} \quad (3)$$

$$Area = \frac{distance(p_2, p_5)}{4}^2 * pi \quad (4)$$

Circularity = distance(p$_1, p_2) + distance(p_2, p_3)$

+distance(p$_3, p_4) + distance(p_4, p_5)$

+distance(p$_5, p_6) + distance(p_6, p_1)$

(5)

Because of the squared term in the denominator, someone with half-open or almost-closed eyes will have a substantially lower pupil circularity value than someone with fully open eyes. The expectation was that, similar to the EAR, when someone is sleepy, their pupil circularity will decrease.

Mouth aspect ratio over Eye aspect ratio

Finally, MOE was included as a new feature. The MOE is the MAR/EAR ratio. The advantage of utilizing this feature is that if the individual's condition changes, EAR and MAR are expected to shift in different ways.

$$MOE = \frac{MAR}{EAR} \quad (6)$$

MOE will be more responsive to these changes than EAR and MAR because it will capture the tiny changes in both EAR and MAR and magnify the changes as the denominator and numerator move in different directions. Our hypothesis was that because the MOE used MAR as the numerator and EAR as the denominator, the MOE would rise as the person became drowsy.While all of these features were obvious, when evaluated with our classification models, they produced mediocre results in the range of 55 to 60percent accuracy, which is only a modest improvement over the baseline accuracy of 50percent for a binary balanced classification problem. Despite this disappointment, we made the most essential discovery: the characteristics weren't incorrect; we were just not looking at them appropriately.

IV. Standardization

The process of rescaling one or more attributes to have a mean value of 0 and a standard deviation of 1 is known as data standardization. This is frequently done on classification jobs where the data has a more gaussian distribution, resulting in improved model accuracy.[1]

$$Z = \frac{x_i - \mu}{\sigma}$$

(7)

V. Feature Normalization

We noticed an unsettling pattern while testing our models with the four fundamental criteria mentioned above. Our model performed poorly when we split the frames in our training and test sets by individuals (i.e. a person in the test set will not be in the training set).

This led us to the conclusion that our model was having trouble with fresh faces, and the main reason for this was because each person's default alert state had distinct key features. That instance, person A may have smaller eyes by nature than person B. Even if person A is aware, a model trained on person B will always predict drowsiness when tested on person A since it will identify a drop in EAR and PUC and an increase in MOE. We believed, based on this observation, that normalizing the traits for each individual would produce superior results, and we were proved true.[**?**]

We utilized the first three frames of each individual's alert video as the baseline for normalization to equalize the characteristics of each individual. Each participant's mean and standard deviation for these three frames were determined and used to normalize each feature independently. Mathematically, this is what the normalization equation looked like:

$$NormalizedFeature_{n,m} = \frac{Feature_{n,m} - \mu_{n,m}}{\sigma_{n,m}}$$

(8)

VI. Feature Importance

Feature Importance refers to methods for calculating a score for each of a model's input features — the scores simply express the "importance" of each feature. A higher score indicates that a certain characteristic will have a greater impact on the model used to forecast a particular variable. It's one thing to build a model, but it's another to comprehend the data that goes into it. Feature importance, like a correlation matrix, helps you to see how characteristics relate to the target variable. It also aids in determining which characteristics are unimportant to the model.[3]

You can use the scores calculated from feature importance to reduce the dimensionality of your model when training it. The higher values are often maintained, whereas the lower scores are usually eliminated since they are unimportant to the model. This not only simplifies the model but also speeds up its operation, thus boosting the model's performance. Feature importance can also help you comprehend and communicate your model to others. You can figure out which characteristics contribute the most to your model's prediction capacity by computing scores for each feature.

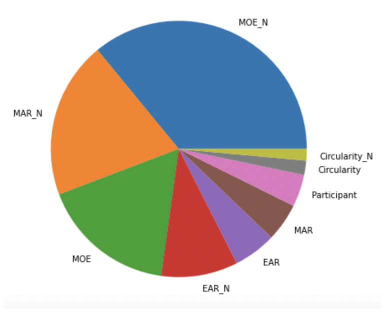


Fig. 4. Feature Importance

We can observe from figure 4 that the most essential features are MAR N, MOE N, and MOE, and that Mouth Aspect Ratio after normalization is the most important feature out of our nine features. This makes sense because we yawn more frequently when we're tired.

Fropm thew figure 5 we can say that logistic regression has accuracy of 86 percent

VII. Precision, Recall and F1-score

The ratio of accurately predicted positive observations to total expected positive observations is known as precision. The better the model is at positive categorization, the higher the precision.

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

(9)

The rate of correct positive case predictions across all positive group samples is measured by recall. The ratio of accurately anticipated positive observations to all positive observations is called recall.[4]

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

The weighted average of accuracy and recall is called F1. As a result, it's more beneficial for evaluating the model's precision and recall.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

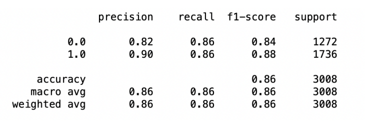
VIII. Proposed Models

In the previous section we have seen pre-processing of data and then in these sections we show what algorithms we introduced and show how these algorithms work on the above pre-processed data. We split the data and turn data into training and testing data and use algorithm to shape the data.

We intended to attempt a variety of modeling techniques after we extracted and normalized our features, starting with the most basic classification models like logistic regression and Naive Bayes and progressing to more complex models with neural networks and other deep learning approaches. It's crucial to keep in mind the balance between performance and interpretability. Although we value high-performing models, we also value interpretability if we are to commercialize this solution and explain its business implications to stakeholders who are unfamiliar with machine learning jargon. We divided our dataset into 17 and 5 videos, respectively, to train and test our models. As a result, our test dataset has 2400 rows and our training dataset has 8160 rows.

Logistic Regression

Despite its name, logistic regression is a classification, not a regression, model. For binary and linear classification problems, logistic regression is a more efficient and easy method. It's a simple classification model that performs well with linearly separable classes. In the industrial world, it is a widely used categorization method. The logistic regression model is a statistical method for binary classification that may be generalized to multiclass classification, similar to the Adaline and perceptron. The logistic regression implementation in Scikit-learn is well efficient and supports multiclass classification tasks.[**?**]



```
              precision    recall  f1-score   support

         0.0       0.82      0.86      0.84      1272
         1.0       0.90      0.86      0.88      1736

    accuracy                           0.86      3008
   macro avg       0.86      0.86      0.86      3008
weighted avg       0.86      0.86      0.86      3008
```

Fig. 5. logistic regression results

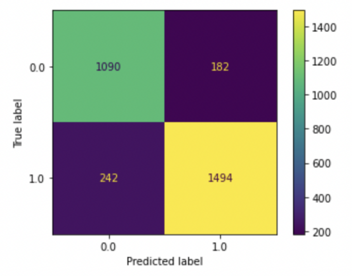
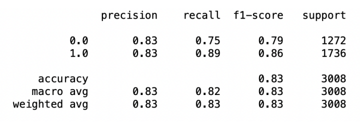


Fig. 6. Confusion Matrix

NaÏve Bayes

NaÏve Bayes is a fantastic example of how the simplest answers are frequently the most powerful. Despite recent developments in Machine Learning, it has shown to be simple, fast, accurate, and dependable. It's been used for a variety of reasons, but it excels in natural language processing (NLP).[6]



```
              precision    recall  f1-score   support

         0.0       0.83      0.75      0.79      1272
         1.0       0.83      0.89      0.86      1736

    accuracy                           0.83      3008
   macro avg       0.83      0.82      0.83      3008
weighted avg       0.83      0.83      0.83      3008
```

Fig. 7. NaÏve bayes results

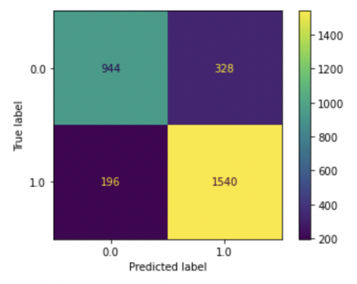
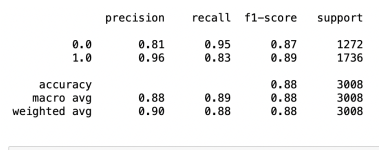


Fig. 8. Confusion Matrix

The Bayes Theorem is the foundation of the Nave Bayes algorithm, which is utilized in a wide range of classification problems.Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. This extension of naive Bayes is called Gaussian Naive Bayes. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

KNN

K-Nearest Neighbour is a Supervised Learning method that is one of the most basic Machine Learning algorithms. The K-NN method assumes that the new case/data and existing cases are similar, and places the new case in the category that is closest to the existing categories. The K-NN algorithm saves all existing data and categorizes additional data points based on their similarity. This means that as fresh data is generated, the K-NN algorithm can quickly classify it into a suitable category. The K-NN algorithm can be used for both regression and classification, but it is more commonly employed for classification. K-NN is a non-parametric method that makes no assumptions about the data. It's also known as a lazy learner algorithm since it doesn't learn from the training set right away; instead, it saves the dataset and uses it to classify. The KNN method simply stores the dataset during the training phase and then classifies new data into a category that is very similar to the old data when it is received.[**?**]



```
              precision    recall  f1-score   support

         0.0       0.81      0.95      0.87      1272
         1.0       0.96      0.83      0.89      1736

    accuracy                           0.88      3008
   macro avg       0.88      0.89      0.88      3008
weighted avg       0.90      0.88      0.88      3008
```

Fig. 9. KNN results

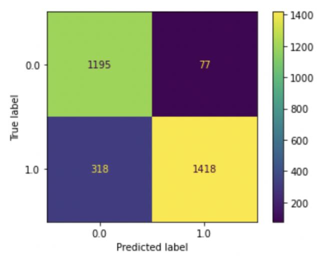
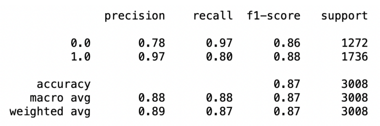


Fig. 10. Confusion Matrix

Multi-layer Perceptron classifier

TensorFlow, Keras, Microsoft Cognitive Toolkit (CNTK), and PyTorch are some of the most prominent deep learning frameworks. Most people are unaware that Scikit-learn, a popular machine learning library, can perform basic deep learning modeling. In this essay, I'll talk about the possibilities and constraints of deep learning modeling in Scikit-learn. I'll also go through two examples of practical implementation. [5]

The output layer doesn't have an activation function. The loss function for regression scenarios is square error, whereas the classification loss function is cross-entropy. It can be used to predict single and multiple target values. Unlike other well-known packages such as Keras, Scikit's MLP version does not support GPU. The settings for each layer, such as distinct activation functions, weight initializers, and so on, cannot be fine-tuned.



```
              precision    recall  f1-score   support

         0.0       0.78      0.97      0.86      1272
         1.0       0.97      0.80      0.88      1736

    accuracy                           0.87      3008
   macro avg       0.88      0.88      0.87      3008
weighted avg       0.89      0.87      0.87      3008
```

Fig. 11. MLP results

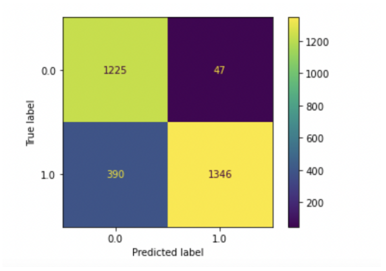
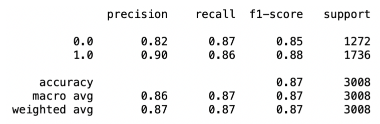


Fig. 12. Confusion Matrix

DECISION TREE

A Decision Tree is a Supervised Machine Learning Algorithm that makes judgments using a set of rules, much like people do. A Machine Learning classification algorithm is designed to make judgements, to put it another way. The model typically predicts the class of a new, never-before-seen input, but the algorithm must decide which class to assign behind the scenes. There are probabilistic classification algorithms, such as Naive Bayes, as well as rule-based approaches. We humans are always making rule-based decisions. You employ a rule-based strategy while you're arranging your next vacation. Depending on how long you'll be on vacation, your budget, and whether or not your extended family will accompany you, you might choose a different location.[7]



```
              precision    recall  f1-score   support

         0.0       0.82      0.87      0.85      1272
         1.0       0.90      0.86      0.88      1736

    accuracy                           0.87      3008
   macro avg       0.86      0.87      0.87      3008
weighted avg       0.87      0.87      0.87      3008
```

Fig. 13. Decision Tree results

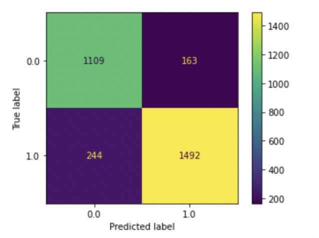
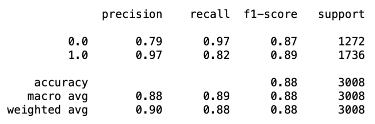


Fig. 14. Confusion Matrix

Because decision trees can handle both classification and regression tasks, they're referred to as the CART algorithm (Classification and Regression Tree). This is a catch-all word for all tree-based algorithms, not just decision trees. But let's concentrate on categorization decision trees. The idea behind Decision Trees is to build yes/no questions using dataset attributes and then split the dataset until all data points belonging to each class are isolated. You're organizing the data in a tree structure with this method. You're adding a node to the tree every time you ask a question. The root node is the very first node. The answer to a question divides the dataset according to the value of a characteristic.

RANDOM FOREST

Logistic regression, support vector machine, naive Bayes classifier, and decision trees are just a few of the classification techniques available in data science. The random forest classifier, on the other hand, is near the top of the classification hierarchy. Discover how individual decision trees are joined to create a random forest and why random forests are so good at what they do. The key is that the models have a low correlation. Uncorrelated models can provide ensemble forecasts that are more accurate than any of the individual predictions, similar to how low-correlation investments (such as stocks and bonds) join together to build a portfolio that is larger than the sum of its parts. The explanation for this amazing effect is that the trees shield each other from their own mistakes.[12]



```
              precision    recall  f1-score   support

         0.0       0.79      0.97      0.87      1272
         1.0       0.97      0.82      0.89      1736

    accuracy                           0.88      3008
   macro avg       0.88      0.89      0.88      3008
weighted avg       0.90      0.88      0.88      3008
```

Fig. 15. Random Forest results

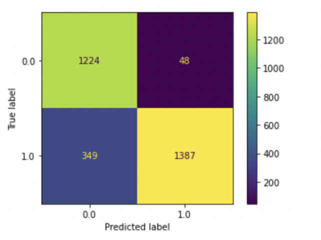
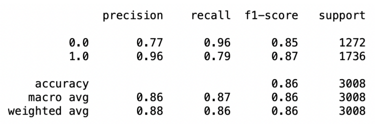


Fig. 16. Confusion Matrix

XG BOOSTING

Extreme Gradient Boosting (XGBoost) is a distributed gradient-boosted decision tree (GBDT) machine learning toolkit that is scalable. It is the top machine learning library for regression, classification, and ranking tasks, and it includes parallel tree boosting. To understand XGBoost, you must first understand the machine learning ideas and methods on which it is based: supervised machine learning, decision trees, ensemble learning, and gradient boosting. Supervised machine learning use algorithms to train a model to detect patterns in a dataset with labels and features, and then to predict labels on fresh dataset features using the learned model.By analyzing a tree of if-then-else true/false feature questions and estimating the least number of questions required to assess the probability of making the correct answer, decision trees generate a model that predicts the label. Decision trees can be used to predict a categorical value or a continuous numeric value using classification or regression.[2]



```
              precision    recall  f1-score   support

         0.0       0.77      0.96      0.85      1272
         1.0       0.96      0.79      0.87      1736

    accuracy                           0.86      3008
   macro avg       0.86      0.87      0.86      3008
weighted avg       0.88      0.86      0.86      3008
```

Fig. 17. XG boosting results

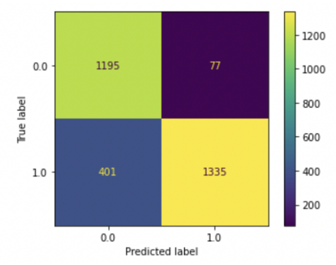


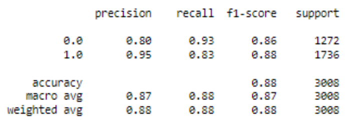
Fig. 18. Confusion Matrix

Convolutional Neural Networks

Convolutional neural networks (CNN/ConvNet) are a type of deep neural network used to analyze visual imagery in deep learning. When we think about neural networks, we typically think of matrix multiplications, but this is not the case with ConvNet. Convolution is a unique technique used. Convolution is a mathematical operation on two functions that results in a third function that expresses how the shape of one is changed by the other. Let's start with the basics, such what is an image and how it is displayed, before moving on to CNN's operation. A grayscale image is a matrix of pixel values with a single plane, whereas an RGB image is a matrix of pixel values with three planes.[12]

Multiple layers of artificial neurons make up convolutional neural networks. Artificial neurons are mathematical functions that calculate the weighted sum of various inputs and output an activation value, similar to their biological counterparts. Each layer creates many activation functions that are passed on to the next layer when you input an image into a ConvNet.

Basic features such as horizontal or diagonal edges are usually extracted by the first layer. This information is passed on to the next layer, which is responsible for detecting more complicated features like corners and combinational edges. As we go deeper into the network, it can recognize even more complex elements like objects, faces, and so on.



```
              precision    recall  f1-score   support

         0.0       0.80      0.93      0.86      1272
         1.0       0.95      0.83      0.88      1736

    accuracy                           0.88      3008
   macro avg       0.87      0.88      0.87      3008
weighted avg       0.88      0.88      0.88      3008
```

Fig. 19. CNN results

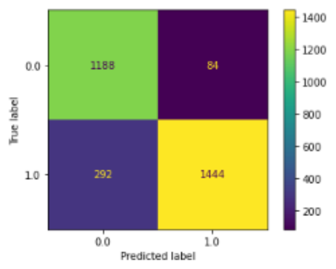
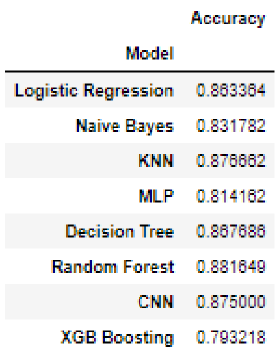


Fig. 20. Confusion Matrix

IX. Results

The below figures are results for the algorithms we have used from this we can say KNN has best accuracy among all other algorithms , so we have used KNN for implementation



Model	Accuracy
Logistic Regression	0.863384
Naive Bayes	0.831782
KNN	0.876662
MLP	0.814162
Decision Tree	0.867686
Random Forest	0.881649
CNN	0.875000
XGB Boosting	0.793218

Fig. 21. Accuracy results for proposed algorithms

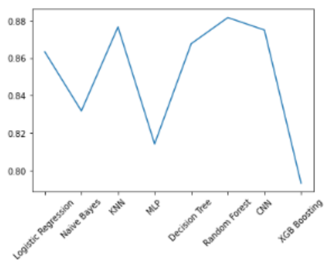


Fig. 22. Graph of Accuracy results

ROC Curves

ROC curves are typically used in binary classification to study the output of a classifier. In order to extend ROC curve and ROC area to multi-label classification, it is necessary to binarize the output. One ROC curve can be drawn per label, but one can also draw a ROC curve by considering each element of the label indicator matrix as a binary prediction (micro-averaging). Another evaluation measure for multi-label classification is macro-averaging, which gives equal weight to the classification of each label.[9]

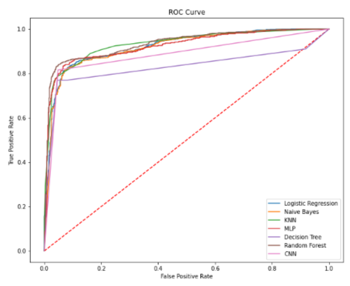


Fig. 23. ROC Curves

The Area Under the Curve (AUC) - ROC curve is a performance statistic for classification issues at various threshold levels. AUC indicates the degree or measure of separability, whereas ROC is a probability curve. It indicates how well the model can discriminate between classes. The AUC indicates how well the model predicts 0 courses as 0 and 1 classes as 1. The higher the AUC, the better the model, by analogy. From the figure 23 we can say that KNN has the higher ROC score which is the best model.

Calibration Cures

Calibration curves are used to determine how well a classifier is calibrated, or how different the probability of correctly predicting each class label are. The average expected probability in each bin is represented on the x-axis. The ratio of positives is represented on the y-axis (the proportion of positive predictions). The ideal calibrated model's curve is a linear straight line travelling linearly from (0, 0).

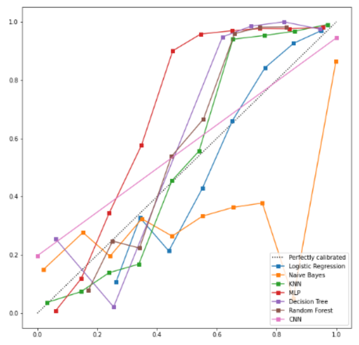


Fig. 24. Calibration Curves

From the above figure 24 we can say CNN(pink) is more linear compared to other curve. From this we can say CNN can predict each class label correctly .

Score of KNN Classifier

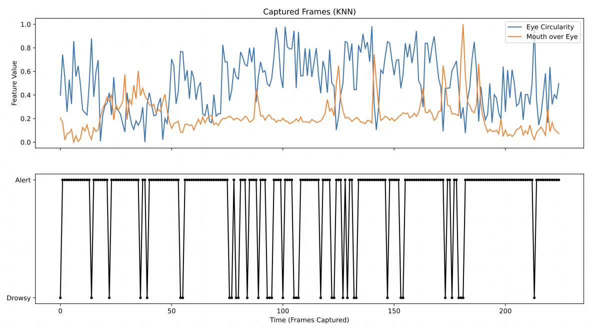


Fig. 25. KNN classifier score

X. Conclusion

We learnt a lot with this process. For starters, simpler models can be as as effective as more complicated models in completing tasks. In our example, the K-Nearest Neighbor model performed similarly to the CNN model in terms of accuracy. However, because we don't want to misclassify people who are sleepy as awake, the more sophisticated model with a lower false-negative rate is ultimately preferable than a simpler model that may be easier to install. Second, we needed to normalize our performance. We realized that everyone's baseline for eye and mouth aspect ratios is different, so we had to adjust for that. Data pre-processing and feature extraction/normalization took up the majority of our effort outside of model running.

XI. Future Work

There are a few things we can do to enhance our outcomes and fine-tune the models in the future. To account for any movement by the person in the video, we must first include distance between the facial landmarks. Participants will not be static on the screen in real life, and we feel that abrupt movements by the participant may indicate tiredness or waking up from micro-sleep. Second, we can improve outcomes by updating parameters in our more complicated models (NNs, ensembles, and so on). Finally, by collecting training data from a bigger sample of people while incorporating additional distinguishing sleepiness signs such as abrupt head movement, hand movement, or even tracking eye movements.

References

[1] Analytics vidhya. https://www.analyticsvidhya.com/blog/2020/04/fe scaling-machine-learning-normalization-standardization/.

[2] Cnn. https://python-course.eu/machine-learning/neural-networks-with-scikit.php.

[3] Feature importance. https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285.

[4] Feature importance. https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285.

[5] Mlp. https://python-course.eu/machine-learning/neural-networks-with-scikit.php.

[6] Naive bayes. https://www.geeksforgeeks.org/naive-bayes-classifiers/.

[7] Randomforest. https://python-course.eu/machine-learning/neural-networks-with-scikit.php.

[8] Reasearch gate. https://www.researchgate.net/figure/Mouth-aspect-ratio-MAR$_{fig}3_{3}$34168196.

[9] Roc. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[10] sleepy foundation. https://www.sleepfoundation.org/drowsy-driving.

[11] Towards data science. https://towardsdatascience.com/drowsiness-detection-with-machine-learning-765a16ca208a.

[12] Xgboost. https://python-course.eu/machine-learning/neural-networks-with-scikit.php.

[13] Venkata Rami Reddy Chirra, Srinivasulu ReddyUyyala, and Venkata Krishna Kishore Kolli. Deep cnn: A machine learning approach for driver drowsiness detection based on eye state. *Rev. d'Intelligence Artif.*, 33(6):461–466, 2019.

[14] Wei Han, Yan Yang, Guang-Bin Huang, Olga Sourina, Felix Klanner, and Cornelia Denk. Driver drowsiness detection based on novel eye openness recognition method and unsupervised feature learning. In *SMC*, pages 1470–1475, 2015.