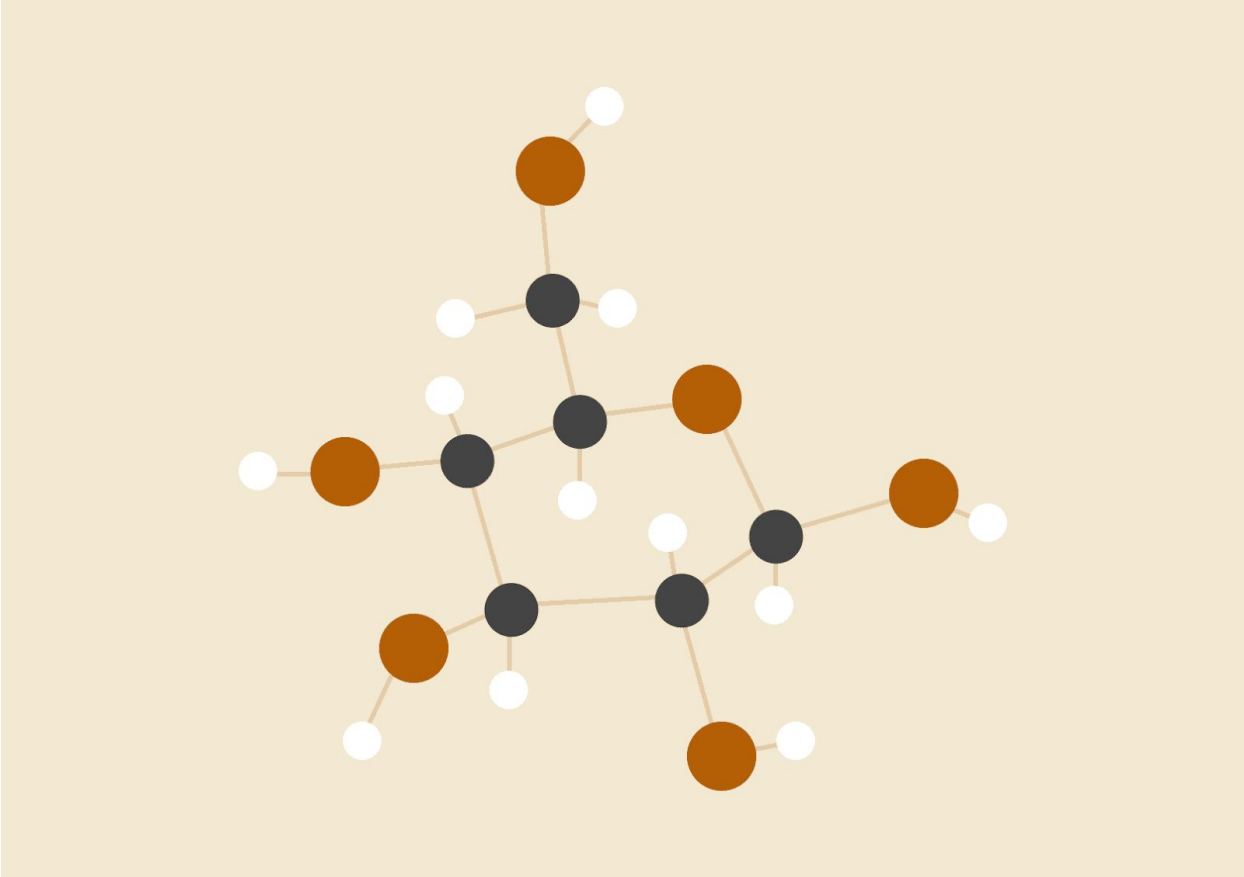
DROWSINESS DETECTION

CS 6375 - Machine Learning



Vineeth Karanam - VXK180045

Lokeswari Umakanthan - LXU190000

PREPROCESSING

The Data contains 120 videos of 60 people, each one taken during the alert and drowsy state of each person. The Videos are taken from UT Arlington Real Life Drowsiness Dataset. We considered the alert and drowsy videos of each person and extracted approximately 100-120 images per video based on 1 image per 5000 frames. The Images are extracted from the videos using OpenCV and Dlib library. The Dlib library is used to extract the coordinate points of the eyes and mouth. The coordinate points can be retrieved in real-time video as well as recorded video using dlib library shape predictor. The get_frontal_face_detector detects the facial coordinate points of a person and the shape_predictor model is passed as a parameter to the shape_predictor of the dlib library. This returns the landmarks of the faces and retrieves 68 coordinate points in the face which give the landmarks like eyes, nose, mouth and the facial boundary.

- The Points from 37 to 42 (Eye) and 49 to 68 (Mouth) are considered for calculating the Eye Aspect Ratio and Mouth Aspect Ratio.
- The Eye Aspect Ratio gives the ratio of height and the width of the eye calculated from the points 37 to 42. Since we are not considering the extreme case of people who have a disability in the eye and normal people will have symmetric eye structure, we are taking the Eye Ratio from only one eye.
- The Mouth Aspect Ratio also gives the ratio of height and width of the mouth from points 49 to 68. The Eye Aspect Ratio and the Mouth Aspect Ratio are inversely proportional.
- The muscles near our eye contracts when a person yawns, so the Eye Ratio and the Mouth Ratio values are inversely proportional. So we consider this attribute as a feature and form a new ratio Mouth and Eye Aspect Ratio.
- The Eye and the Mouth dimensions did not provide more variations from person to person in the dataset, so we are considering the eye pupil circularity which gives the more accurate results for differentiating each person.
- The Eye Ratio, the Mouth Ratio, ratio of eye with mouth and pupil circularity are considered as features for training and testing the model. The Eye Ratio, Mouth Ratio, Eye with Mouth Ratio and the pupil area for being alert and drowsy may vary from person to person.
- So we are normalizing the 4 features Eye Ratio, Mouth Ratio, Eye with Mouth Ratio and Pupil Circularity based on each person. The first 10 frames of each person's alert video at an interval of 10 frames per image is considered as an

normalizing criteria.

- For normalizing we are calculating Z-score and standardizing each feature example to differentiate the alert and drowsiness based on their own normal values.
- The mean and the standard deviation of the first 10 frames at 10 frames per image interval is taken to normalize the each feature example obtained for that person's alert video and drowsy video.
- The formula for Z-score calculation = (Each feature example of person) - (mean of feature of that person in alert) / (standard deviation of feature of that person in alert).
- These additional normalized values of each feature are used for training and testing the model. So totally 8 features Eye Ratio, Mouth Ratio, Eye with Mouth Ratio, Pupil Circularity, Normalized_eye_ratio, Normalized_Mouth_ratio, Normalized_Eye_Mouth_ratio and Normalized_pupil_circularity is feeded to the model.

ALGORITHMS USED

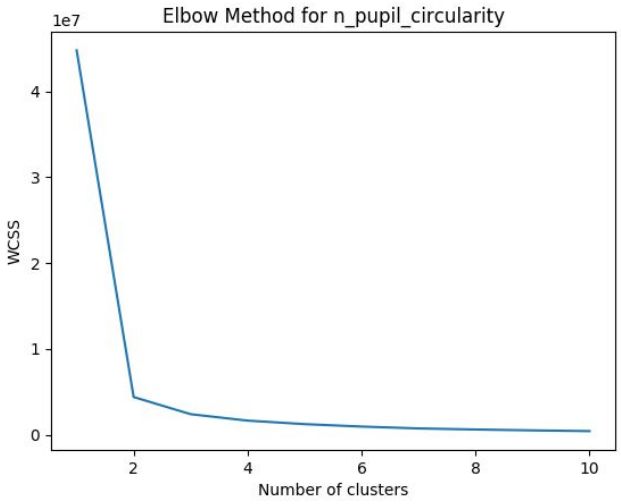
1. K - Nearest Neighbors (Own implementation)
2. Gradient Boosting (Own implementation)
3. Bagging and ADABoost (Own implementation)
4. XGBoost (Standard implementation)
5. K - Means Clustering (Standard implementation) for preprocessing

K-MEANS CLUSTERING

As we are analysing a classification problem, we are using algorithms like Gradient Boosting, Decision Trees (with ensemble methods) etc. Since we haven't learnt how to build regression trees, we need the input data to be discrete and categorical. For discretizing the data, we need to first determine the number of bins for each feature to be categorized into. For this we used K-means clustering Within Cluster Sum of Squares to find out the variation of the number of clusters based on the Within Cluster Sum of Squares of each feature data. The best number of bins has been determined by analyzing the Within Cluster Sum of Squares graphs using the elbow method. So for each feature we get a different number of best clusters i.e. the number of categories each feature to be discretized into.

To check if the elbow method yields a best output, we ran a different combination of the

categories for each feature and trained and tested the model. This yielded more accuracy when features are split on a specific number of categories which matches with the elbow method of K-means clustering.



K NEAREST NEIGHBORS

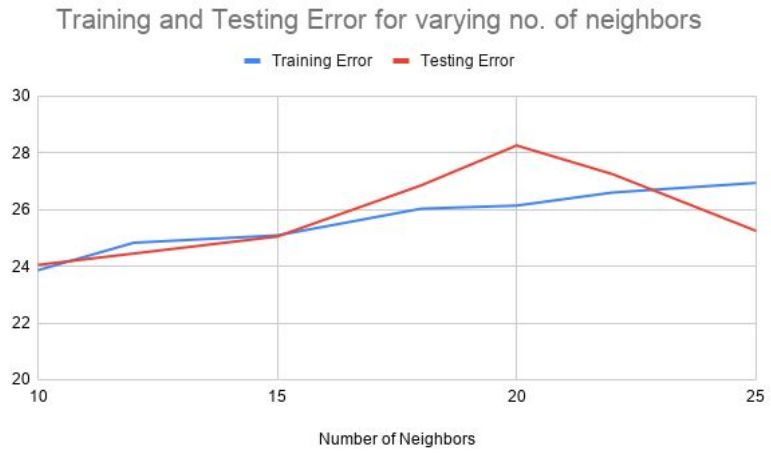
Since we are working on a real time data, we thought K Nearest Neighbor would help in instance based learning of the data. We trained and tested the data in our own implementation of the algorithm. K Nearest Neighbors being a lazy algorithm, trains the model without explicitly framing a function for predicting the test examples. Rather it memorises the data and categorizes the test example to an output class based on that. Since every person's Eye Ratio, Mouth Ratio, Eye with Mouth Ratio and Pupil Circularity varies from person to person, instead of making the model learn explicitly, we tried using KNN to predict the test example in an instance based approach.

Another reason for choosing KNN is that it uses a non-parametric approach i.e. it does not make any assumptions on the data. The model should not make any assumptions on the data retrieved from the real world, because the real world data may vary from person to person and making assumptions may cause data misrepresentation and loss. Since it is not making any assumptions, it is also robust to K value.

- Increase in K value increases the accuracy to a certain level, after a threshold it starts to overfit the model. So K value must be identified in such a way that it should not overfit or underfit the model.
- We have used the K-value from the range (10,12,15,18,20,22,25). The results obtained are in the form of normal distribution, when k value = 10, the error is

lower with lower k-value and it increases with increase in k-value. We have used the data with real values without discretizing the values of the features, since we would be able to train and test KNN with real values which yielded better results than for discretized data.

- After a threshold of k-value = 20 , the error reduces at lower k-value and the increase in K value overfits the model. This shows that for k=10 to 18 the training error varies from 23.68% to 26.03% and testing error varies from 24.02% to 26.85%. At K=20 the training and testing error are 26.14% and 28.26%.
- After k=20, for k =22,25 the training error increases from 26.60% to 26.94% and testing error reduces from 27.25% to 25.25%. The increase in testing error and simultaneous reduction of training error leads to overfitting of the model. These values are plotted in the graph and visualized as below.



BAGGING AND ADABOOST

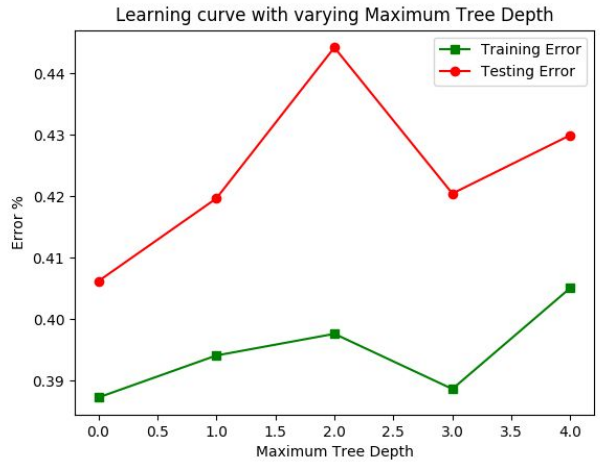
We are classifying if the person is in Alert State or Drowsy State from the Drowsiness Detection Dataset using the Bagging and AdaBoost of our own implementation of the algorithm on discretized data. For classification problems it is the best solution to use a decision tree with ensemble methods. We have used Bagging by varying the range of Maximum Depth and the Number of Trees for Bagging. For ADABoost we varied the range of the Number of Stumps(Trees with one depth).

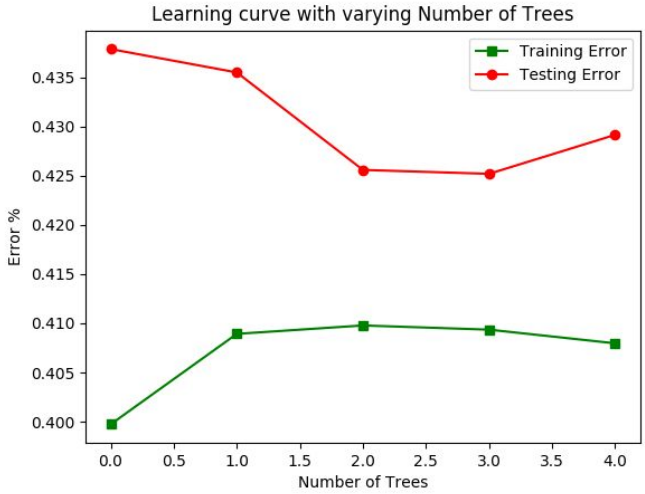
Bagging reduces the variance whereas ADABoost reduces the bias, this increases the accuracy of the many weak learners. Bagging performs comparatively higher than the

ADABoost, since we are providing discretized data to the model which causes data loss to the actual dataset.

Since our Bootstrap replicate approximation is correct and there is no way of the replicates becoming worse with outliers and noise, the bagging reduces variance without changing the bias and produces better accuracy than ADABoost. Boosting normally has a high risk of overfitting the data. Since our data is more generalized by discretized data, it is prone to overfitting. ADABoost considers previous misclassifications to reduce the bias, but in our dataset there may be unpredictable misclassifications since the features are discretized and vary from person to person.

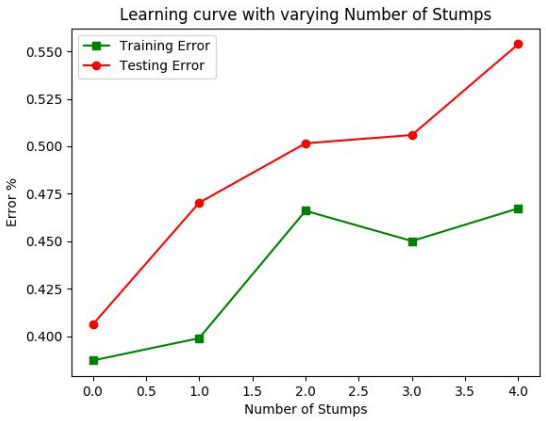
Bagging





BAGGING OBSERVATION : The increase in the maximum tree depth from 1 to 4, increases the training error and testing error till depth =2 and reduces at depth =3. Increase in the number of trees till 2 reduces the error rate and again increases from number of trees 3 to 5. This shows that at depth = 3 and number of trees = 2. Since we are using 8 columns as features we got better accuracy at a lower range of values.

ADABoost



ADABOOST OBSERVATION : The Increase in the number of stumps increases both the training and testing accuracy. Since our data is discretized, there may be higher chances of overfitting the data. Hence AdaBoost does not provide better results than bagging.

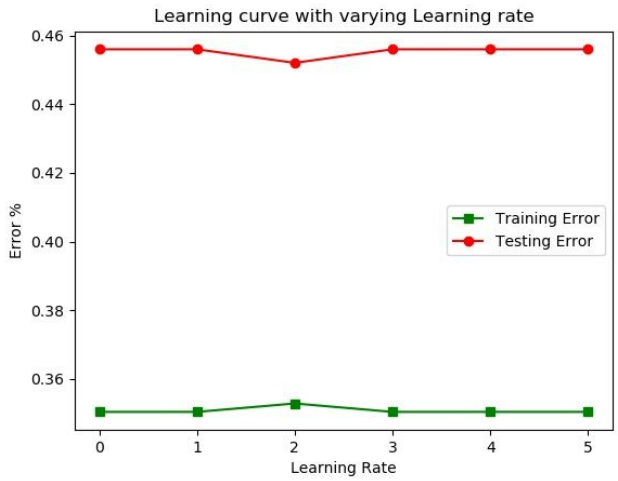
GRADIENT BOOSTING

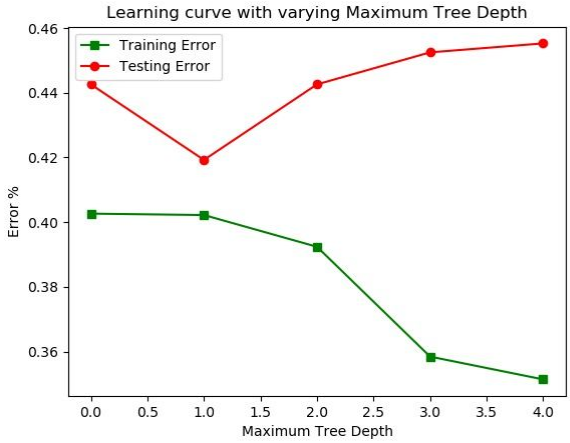
We have trained and tested the discretized data in the our own implementation of gradient boosting algorithm.

Since this is a classification problem, we thought it would be a better solution to use decision trees built upon the gradient boosting methods. In AdaBoost the accuracy decreases because adaboost does not reduce variance and overfitting, we try to reduce the overfitting by using gradient descent and boosting i.e. gradient boosting.

The gradient boosting in the function space is the same as the gradient descent in the prediction space, since our dataset deals with the real time we do not want the model to vary in the prediction space than the function space. These kinds of variations would not occur if the data is normalized else this causes overfitting of the model. Even though we are normalizing the data and providing as input, the variations in the prediction space

may affect the new dataset of an individual person.

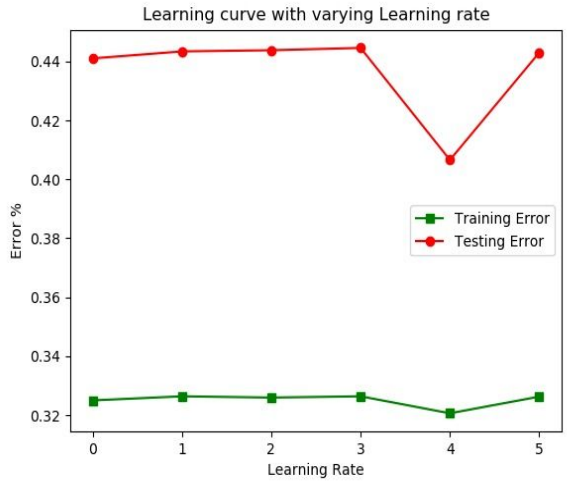


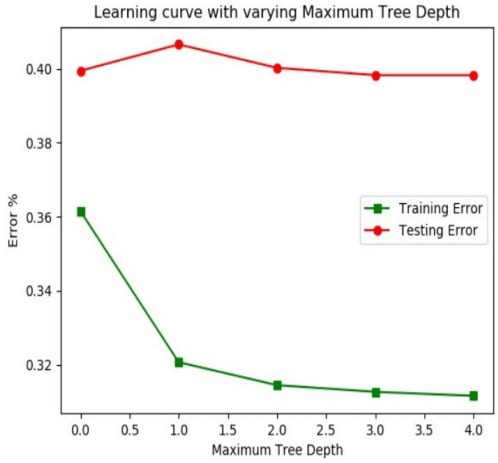


GRADIENT BOOSTING OBSERVATIONS : There are no huge variations in the training error and testing error during the increase in the learning rate. But the increase in the depth of the tree reduces the error rate initially at tree depth = 1 and overfits the data because of the increase in testing error.

XGBOOST

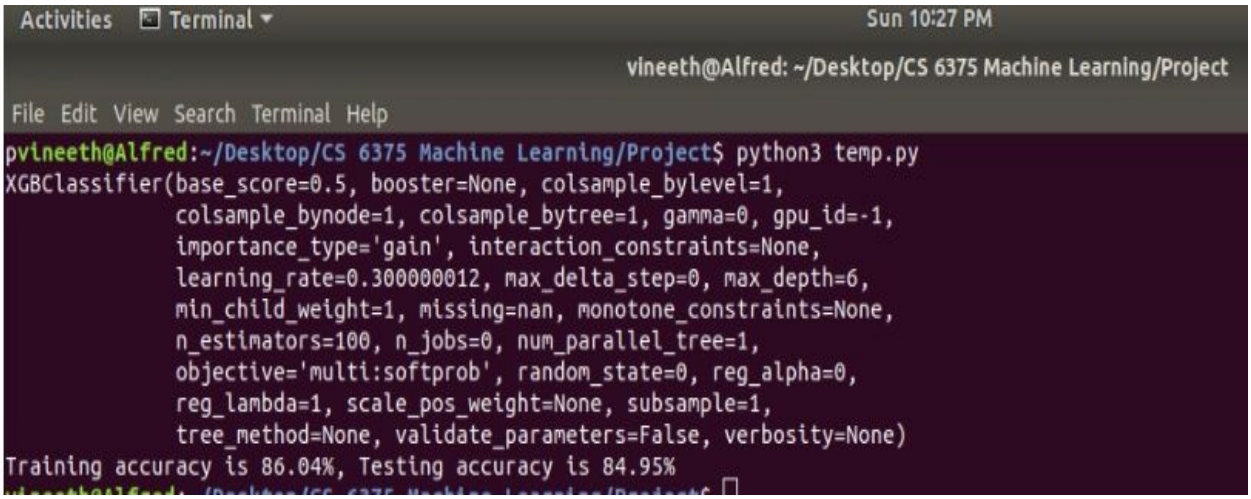
Since our results in AdaBoost and Gradient Boosting are not satisfying, we chose Extreme Gradient Boosting. Extreme Gradient boosting randomization reduces the correlation between the trees.





OBSERVATION: The increase in the learning rate decreases error rate initially and then increases after the learning rate of 4. The maximum tree depth at 2 provides better results than other points.

The below image is the result of the XGboosting of sklearn using the default features. This provides an accuracy of 80% since the data provided to it is the real data and not discretized data. This shows that the model provides best results at learning rate=0.3, n-estimators = 100 and depth =6.



DATA

- The data has been taken from the UT Arlington Real Life Drowsiness Dataset available for 60 individuals Real Life Drowsiness Dataset available here.

E.A.R.	M.A.R.	M.O.E.	Pupil Circularity	Features normalized using z-score				Status
				E.A.R.	M.A.R.	M.O.E.	Pupil Circularity	
3.0271	1.0613	0.3506	0.0602	5.5318	0.9901	-1.9713	-2.5923	0(Alert)
2.4335	1.0102	0.4151	0.0848	-1.3408	-0.2176	0.5785	1.7846	1(Drowsy)

CONCLUSION

KNN runs superior over all the tree based algorithms used as the discretization of the data leads to loss of information. Regression trees need to be used. Majority voting can be used to mitigate the effect of outliers but the algorithms discussed above will not be as good as LSTMs in terms of their results over real time data. Leave one out strategy gives the evaluation of your model.

REFERENCES

1. Ghoddoosian, R., Galib, M., & Athitsos, V.:A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 0-0 (2019)