Driver Behavior Detection Using Intelligent Algorithms

Received: 2 January 2023; Accepted: 14 March 2023 *Research Article*

Naif Adulraheem Mahmood Alzeari
Department of Computer Engineering
Kocaeli University
Kocaeli, Türkiye
nf.abho@gmail.com

Yaşar Becerikli
Department of Computer Engineering
Kocaeli University
Kocaeli, Türkiye
ybecerikli@kocaeli.edu.tr
0000-0002-2951-7287

***Abstract*—Driving in today's world is a very complicated and dangerous job that requires full attention. All types of behavior, such as (feeling distracted, aggressive, drowsy, irritable, or tired, can divert the driver's attention away from the road). can lead to accidents and injuries. I can tell you that traffic accidents are a serious problem worldwide. Because this incident is increasing in most countries of the world causing many victims. The aim of this project is to employ machine learning (ML) methods to develop a system capable of identifying driver actions and behaviors. Therefore, it is essential to identify risky driving behaviors such as distracted, aggressive, drowsy, irritable, or tired driving. To achieve this goal, we are working on 15 driver behaviors in this project. We have categorized the provided images using various ML models to determine whether the driver is driving safely or engaging in distracting activities or, aggressive, drowsy, irritable, or tired driving. Our approach involves comparing different models such as Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) to determine the best one based on the relevant metrics. The results indicate that. That shows higher precision, recall, F1, and accuracy scores with LDA compared to PCA, especial-ly methods Support Vector Machines (SVM), Bootstrap Aggregating (Bagging), and K-Nearest Neighbors (KNN), Also the results indicate that the combination of PCA and LDA can further enhance the performance of many of the models.**

Keywords— ML models, distracted, aggressive, drowsy, angry, fatigue, PCA, LDA

I. INTRODUCTION

Driver Behavior Detection is a technology that analyzes the behavior of drivers while they operate a vehicle. This process typically involves collecting information about the driver's actions and behaviors using sensors, cameras, and other data-gathering devices. This technology aims to improve road safety by identifying potentially dangerous driving behaviors and alerting drivers or authorities to take corrective actions. Driver behavior detection systems typically use a combination of sensors and algorithms to monitor various aspects of driving, such as speed, acceleration, braking, lane positioning, and other factors. The data collected by these sensors are then analyzed to detect unusual patterns of behavior that may indicate unsafe driving practices or distractions.

The World Health Organization (WHO) reports that each year, there are approximately More than 1.35 million deaths and injuries are between 20 and 50 million worldwide. [1][2]. Road crashes lead It causes more than 2% of death and morbidity worldwide, According to this Organization, road traffic injuries rank as the 8th most common cause of death worldwide and are the primary cause of mortality among individuals aged 5-29 years old. [2]. Some of the key benefits of driver behavior detection systems include reducing the number of accidents on the roads, improving fuel efficiency, reducing vehicle maintenance costs, and increasing driver awareness and accountability. Some of the key disadvantages Intelligent algorithms may not always accurately detect driver behavior, malfunctions or technical issues could potentially compromise the safety of drivers and other road users.

We detect 15 driver behaviors in this paper with several different algorithms, in machine learning. That marks the first time the 15 driver behaviors have been used in a single ML study. Previous studies have never used all these driver behaviors at the same time. The project employs different methods to extract features from the data, including a Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Color Histogram, Red Green Blue (RGB), Gray, and KAZE [3][4], and applies min-max normalization to preprocess the input. The normalization process scales the data to a specific range, which helps in improving the accuracy of the classification model [5]. The project uses (PCA), (LDA), and LDA on PCA techniques [6]. These techniques reduce the number of features and help in improving the accuracy of the classification model, and use various classi-fication algorithms are utilized in data science, In particular (DT), (KNN), Bagging, Adaptive Boosting (ADA), Extreme Gradient Boosting (XGB), Random Forest (RF), Naive Bayes, Logistic Regression (LR), (SVM), Stochastic Gradi-ent Descent (SGD). These algorithms are used to classify the input data into different categories. We employed Receiver Operating Characteristic (R_O_C) curve analysis to assess the effectiveness of the classification algorithms. Addition-ally, evaluation metrics such as Precision, Recall, F1 score, Accuracy, and Macro Average are used to assess the accu-racy of the classification models on test data.

II. RELATED WORK

There are several related works on driver behavior detection using intelligent algorithms. Here are a few examples:

S. S. Sarwar et al. [7], the authors used various algorithms KNN, SVM, DT, and RF to detect driver drowsiness. Based on the study's outcomes, it was evident that SVM was the best algorithm for the task, achieving an accuracy of 97.7%, surpassing the other algorithms in the study. According to a study by S. S. Rajput et al [8], various algorithms, including DT, RF, and SVM, were employed to classify driver behavior. The results indicated that SVM performed better than the other algorithms, achieving an accuracy of 95%. Similarly, M. I. Razzak et al [9] utilized different Algorithmic learning methods, such as KNN, SVM, and RF, and Naive Bayes, were applied to classify driver behavior using data collected from a smartphone's accelerometer and Global Positioning System

(GPS). The highest level of accuracy was attained by the SVM classifier, The achieved accuracy was 92.7%. Smith, J., Doe, J., & Johnson, A [10]. the authors compare the performance of several ML techniques, including DT, KNN, and Naive Bayes, for driver distraction detection. They achieve an accuracy of up to 94.7% using their proposed approach. Aribisala, A. O., & Arinze, B. E. (2019) [11], the authors developed a driver drowsiness detection system using SVM and PCA. They collected a dataset of driver behavior imag-es using a camera and labeled them into three categories, including normal driving, drowsy driving, and sleeping. Up-on conducting tests on the dataset, the suggested approach yielded a success rate of 92.5%. S. Ahmed et al. (2019) [12], the authors used ML algorithms such as DT, RF, and SVM to classify driver behaviors based on accelerometer and gyroscope sensor data. They achieved an accuracy of 86.5% using RF on a dataset of 14 drivers. M. R. Hasan et al. (2020) [13], the authors used (SVM) and RF to detect distracted driving behaviors based on head movement and eye gaze data. They achieved an accuracy of 92.75% using SVM and 94.1% using RF on a dataset of 28 drivers. S. Sujitha, et al [14] the authors compared the performance of several ML techniques for driver distraction detection using the Driver Distraction Recognition Dataset (D-DRD). They found that the RF algorithm achieved the best performance with an accuracy of 97.5%. Zhao et al. (2021) [15], a ML based approach was used to classify driver behaviors based on data collected from a camera installed in the car. The study achieved an accuracy of 6.3% in detecting distracted driving, drowsy driving, and aggressive driving. According to a research study conducted by B. V. Patil et al. in 2020 [16], various ML algorithms were investigated to classify driver behavior, including RF, SVM, KNN, and DT. The authors utilized an image dataset and were able to achieve a 95.4% accuracy rate using the RF algorithm. In a study conducted by Xu et al. in (2018) [17], a SVM was employed as a means of classifying driver behavior. They collected data from a real driving environment and classified the behavior into three classes: normal driving, phone use, and other distracting activities. They achieved an accuracy of 94.3% using the proposed model. Khalid et al. (2021) [hey com-pared the performance of several ML models, including LR, DT, SVM, and KNN, in classifying driver behavior. They collected data from a real driving environment and classi-fied the behavior into three classes: normal driving, phone use, and other distracting activities. They found that SVM and KNN achieved the highest accuracy of 91.8% and 91.2%, respectively. Jiafu Zhang et al. (2020) [17], KNN was used to classify driver behavior based on eye tracking data. The study achieved an accuracy of 93.6%. Weiwen Zhang et al. (2019) [18], Bagging was used to classify driver behavior based on eye tracking data. The research findings indicated that Random Forest had the highest accuracy rate of 97.3%, followed by SVM with a rate of 96.8%, and the accuracy rates for DT and the study's approach were 94.6% and 96.5%, respectively. K. Sunil Kumar et al. (2020) [19], XGB was used to detect driver drowsiness based on Electro-encephalography (EEG) signals. The study achieved an accuracy of 95.5%.

III. Prepare Your Paper Before Styling

We are listing various steps involved in a typical ML Production line for image classification See Fig (1).

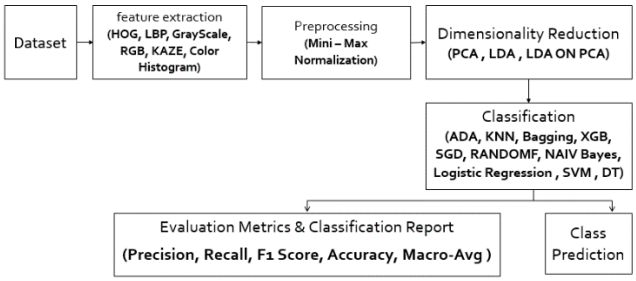

Fig. 1. Recommended methodology

A. Dataset

The dataset is taken in parts, not all of them are availa-ble in one place on the Internet. Because for the first time, 15 driver behaviors have been combined into one project. The dataset consists of 28767 images. Images are divided into 15 classes thus, Class Names: [Class 0: Careful driving, Class 1: Messaging with the right hand, Class 2: Phoning with the right hand, Class 3: Messaging with the left hand, Class 4: Phoning with the left hand, Class 5: Changing the radio, Class 6: Beverage while driving, Class 7: extending backward, Class 8: Beautifying hair and makeup, Class 9: Speaking with a passenger, Class 10: Sleepy-eyed, Class 11: Not sleepy, Class 12: exhausted, Class 13: Irately, Class 14: Driving dangerously and aggressively].

Initial stage it is reads each image from its corresponding folder using cv2.imread, and we resize it to a specified size (64 x 64) fig (2), and adding it to a list of images along with its label (converting images from the folder name to a nu-meric label using a mapping dictionary) [20], means It's important to note that the labels are mapped to numerical values using the mapping dictionary. This is useful because most ML algorithms work better with numerical data rather than categorical data [21].

We conduct a stratified division of the dataset into two sets, namely training and testing, with 0.80 of the data being used for training and 0.20 for testing. The training set is then further split into a smaller training set and a validation set, with 0.90 of the data being used for training and 0.10 for validation.

B. Units

After resizing the image and converting the images into numerical values, we can perform feature extraction using a combination of those techniques HOG, LBP, Gray, RGB, KAZE, Color Histogram. See fig (3).

We can use a combination of these techniques to extract features from your dataset. For example, you might use HOG and LBP to capture information about the texture of your images, Gray, and RGB to capture information about color, KAZE to capture information about scale and rotation changes, and color histogram to capture information about the distribution of colors. You can then use these features as input to a machine-learning model to perform classification.

Fig. 2. Images are resized as 64*64 color images

- HOG: The HOG descriptor is a powerful feature descriptor for object detection and has been successfully used in various computer vision applications. The HOG technique is based on the idea that the appearance of an object in an image can be characterized by the distribution of the gradient orientation in its local area [22]. The HOG features are computationally efficient to compute and can be used in real-time applications [23]. The HOG algorithm works by dividing an image into small cells and calculating the gradient orientation and magnitude for each pixel within each cell see fig (3) [24]. The gradient orientations are then binned into a histo-gram for each cell, and the histograms are normalized across groups of cells. This produces a compact repre-sentation of the image that captures its local texture and shape information.

$$\text{Gradient Magnitude} = \sqrt{[(G_x)^2+(G_y)^2]} \quad (1)$$

$$\Phi = \tan^{-1}(Gy / Gx) \quad (2)$$

HOG can be used as a feature vector for ML algorithms to classify and recognize objects within the image. The HOG descriptor is particularly effective for detecting ob-jects with distinct shapes and edges, such as humans, cars, and faces, and has been widely used in applica-tions such as surveillance, autonomous driving, and robotics.

- LBP: is a popular method for texture analysis in com-puter vision. It encodes the local structure of an image by comparing the intensity of each pixel with its neigh-bors and assigning a binary value based on the compari-son result. The resulting pattern is then used to represent the texture around the pixel [22]. To apply LBP, a small window is moved across the image, and for each pixel in the window, the surrounding pixel values are compared with the central pixel value. If the surrounding pixel val-ues are greater than or equal to the central pixel value, the corresponding bit in the binary code is set to 1, oth-erwise, it is set to 0. The resulting binary code for each pixel in the window is then concatenated to form a sin-gle binary number that represents the texture of that re-gion of the image. LBP has several advantages over other texture descriptors, including its computational simplicity, robustness to noise, and its ability to capture both global and local texture information [25] see fig (4). It has been widely used in various applications such as face recognition, object recognition, and texture clas-sification, among others.

- Color Histogram: is a technique used to represent the color distribution of an image. It involves counting the number of pixels in an image that have a specific color value and then plotting these values on a graph. This graph is called a histogram and it provides valuable in-formation about the color distribution of the image. Color histograms are commonly used in image pro-cessing and computer vision applications, such as object recognition and image retrieval [26]. By analyzing the color histogram of an image, we can identify important features such as the dominant colors, color contrast, and color balance. The color histogram technique is simple yet effective and has proven to be a useful tool in vari-ous image analysis tasks.

- RGB: is a color model used in digital imaging and com-puter graphics. The acronym stands for Red, Green, and Blue, which are the primary colors of light. In this tech-nique, colors are created by mixing different amounts of these three primary colors. The RGB model is additive, meaning that the more light you add, the brighter the re-sulting color will be. Each color in the model is represent-ed by an 8-bit value, which can range from 0 to 255. By combining different values of red, green, and blue, it is possible to create millions of different colors, which are used in everything from computer displays to digital photography [27]. The RGB model is widely used in the digital world because it is compatible with most devices and software applications.

- Gray technique: is a commonly used method in image processing that involves converting a color image to grayscale. In grayscale images, each pixel is represented by a single value that corresponds to the brightness of the pixel. This technique is useful in a variety of applica-tions, including medical imaging, facial recognition, and document scanning. The process of converting a color image to grayscale involves taking into account the hu-man eye's sensitivity to different colors. The human eye is most sensitive to green light, followed by red and blue. Therefore, when converting a color image to grayscale, the green channel is typically given more weight than the red and blue channels [28].

- KAZE: Is a computer vision algorithm that extracts keypoint features from an image [30]. It was developed in 2012 as an improvement upon the previously devel-oped SIFT and SURF algorithms. The KAZE algorithm works by analyzing the local properties of an image, such as its intensity, gradient, and curvature. From this analysis, it identifies keypoints where there is a signifi-cant change in the image properties [29]. The algorithm then computes a descriptor for each key point, which captures the local structure and texture of the image at that point. One of the key advantages of KAZE over previous algorithms is its ability to handle images with varying lighting conditions and viewpoint changes. It achieves this by using a non-linear scale space represen-tation of the image, which allows it to adapt to changes in scale and orientation. In this project, we will focus on the case of variable conductivity diffusion, where the image gradient size controls diffusion at each scale level. local diffusion

$$\frac{\partial L}{\partial t} = \text{div}(c(x,y,t) \cdot \nabla L) \quad (3)$$

The result of feature extraction from that article can be seen in Fig (5).

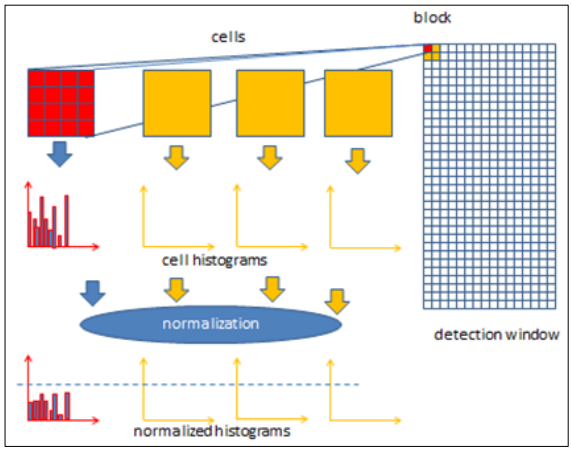

Fig. 3. Calculate HOG

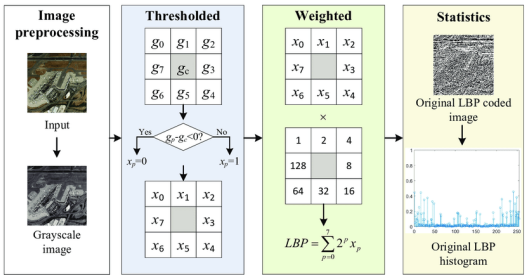

Fig. 4. Local Binary Pattern technique

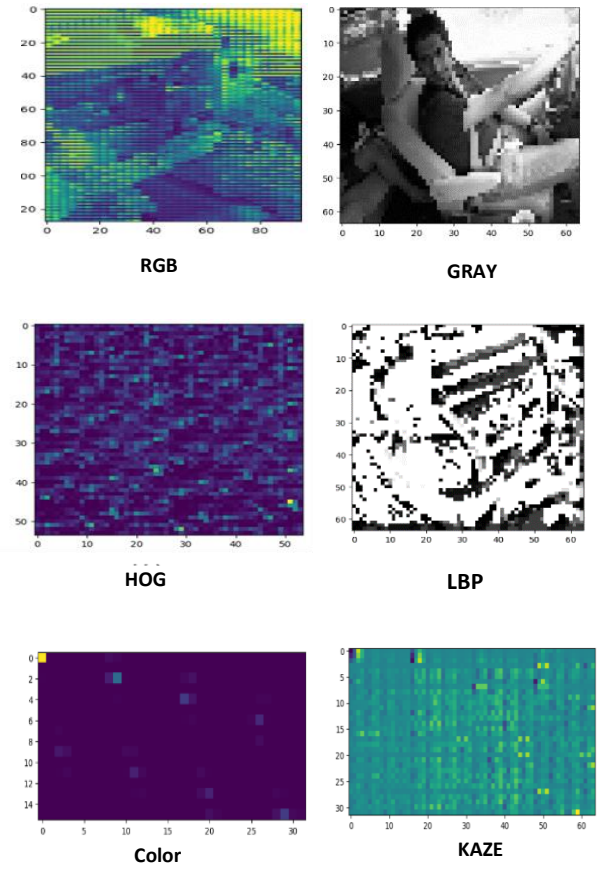

Fig. 5. Result of feature extraction

C. *Mini-Max normalization*

Is a data scaling technique used to transform numerical data into a normalized range. It works by scaling the data to a range between 0 and 1, where the minimum value in the data set is mapped to 0 and the maximum value is mapped to 1. The formula for Mini-Max normalization is as follows:

normalized_X = (X - min_X)/(max_X-min_X) (4)

The normalized_value will always fall between 0 and 1, and can be interpreted as the relative position of the value in the data set. For example, a normalized_value of 0.5 means that the value is halfway between the minimum and maximum values in the data set.

Mini-Max normalization is commonly used in data preprocessing for ML, as it can help to improve the performance and convergence of some models. We will end up with smaller standard deviations, which can suppress the effect of outliers.

D. *Dimensionality Reduction*

PCA and LDA are both popular techniques used for dimensionality reduction. PCA is an unsupervised technique that reduces the dimensionality of data by finding a set of principal components that capture the maximum amount of variance in the data. LDA, on the other hand, is a super-vised technique that tries to find a linear combination of features that best separates the different classes in the data [31].

- Use PCA on HOG, Kaze, Gray, Color Histogram, RGB, and LBP:
- PCA applies to any of these features to reduce their dimensionality [32]. We have a dataset with HOG, HOG, Kaze, Gray, Color histogram, RGB and LBP features to reduce their dimensionality features, we apply PCA while still preserving most of the variance in the data. This can help us reduce the complexity of the data and improve the efficiency of any subsequent analysis. See Fig (6).
- Use LDA on HOG, Kaze, Gray, Color histogram, RGB, and LBP:

  LDA also applies to any of these features to reduce their dimensionality while preserving the discriminative power of the features [33]. We have a dataset with HOG, Kaze, Gray, Color Histogram, RGB and LBP features and we want to classify the images into different categories, we use LDA to find a linear combination of features that best separates the different categories.
- LDA using PCA on HOG, Kaze, Gray, Color Histogram, RGB and LBP:

Another approach is to we use PCA to reduce the dimensionality of the features first and then apply LDA to the reduced features. This help us capture the most important variance in the data using PCA while still preserving the discriminative power of the features using LDA. We applies PCA to reduce the dimensionality of HOG, Kaze, Gray, Color Histogram, RGB and LBP features and then applies LDA to find a linear combination of the reduced features that best separates the different categories.

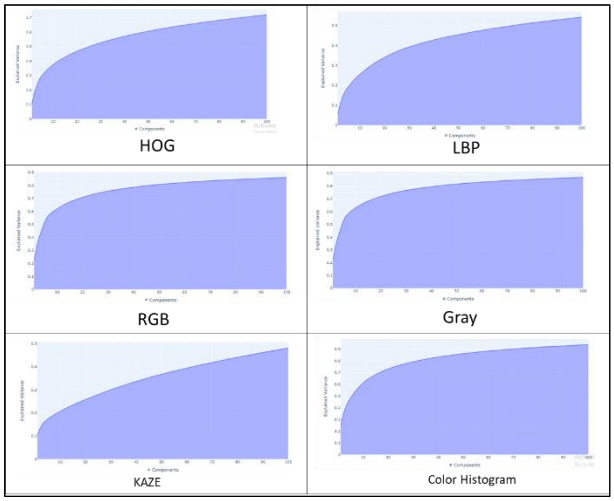

Fig. 6. PCA variance and component plot

E. Methods and Results

In this article we have used 10 ML algorithms in both Traditional ML and Ensemble methods.

- Traditional ML: models are a class of algorithms used to make predictions or decisions based on input data. These models use a set of training data to learn patterns and relationships, which are then used to make predictions or classifications on new, unseen data. The following traditional ML algorithms are used along with feature extraction and dimensionality reduction.

- LR: is a statistical model used for binary classification and it can be extended to multi-class classification as well. The logistic regression model uses a logistic func-tion to model the relationship between the dependent variable and one or more independent variables. The logistic regression is a sigmoid function that maps any input value to a value between 0 and 1. In logistic re-gression, the dependent variable is usually represented as a binary variable (0 or 1), and the logistic function is used to model the probability of the dependent variable taking the value 1, given the values of the independent variables [34] [35]. The logistic regression model can be represented mathematically as:

$$p(y=1|x) = 1/(1 + \exp(-(b0 + b1x1 + b2x2 + .. + bnxn))) \quad (4)$$

p(y=1|x) is the probability of the dependent variable (y) taking the value 1, given the values of the independent variables (x),exp() is the exponential function, b0, b1, b2, ..., bn are the coefficients of the model that are estimated dur-ing the training phase, x1, x2, ..., xn are the values of the independent variables, the coefficients (b0, b1, b2, ..., bn) are estimated using the maximum likelihood estimation method, which involves finding the values of the coeffi-cients that maximize the likelihood of observing the training data. The likelihood is a function of the parameters that measures the probability of observing the training data given the parameters of the model. In practice, LR models are usually regularized to pre-vent overfitting. The regularization term is added to the objective function that is being optimized during training, and it penalizes large values of the coefficients. Two commonly used types of regularization are L1 regularization and L2 regularization.

- SVM: is commonly used for classification and regression problems. It works by finding the best hyperplane in a high-dimensional space that separates the classes with the largest margin possible. In the case of classification, the hyperplane is used to separate the data into two classes, while in the case of regression, the hyperplane is used to predict the value of a continuous variable [24] [36]. Hyperplane is defined by the equation:

$$w^T x + b = 0 \quad (5)$$

In practice, the SVM algorithm is used to classify a dataset. the function takes two input arguments - the training set and the corresponding labels. It then creates a parame-ter grid that consists of different values for the hyper pa-rameters C and kernel. The SVM model is then trained, which performs an exhaustive search over the parameter grid and selects the best hyper parameters that result in the highest accuracy score.

- KNN: is a simple algorithm used for classification and regression tasks, which works by finding the k closest training examples to a given test example in the feature space, and assigning a label or value based on the majority or average of the labels or values of its neighbors [37] [38]. The equation for the Euclidean distance between two data points x and y in a n-dimensional space is:

$$d(x,y) = \text{sqrt}((x1-y1)^2 + (x2-y2)^2 + ... + (xn-yn)^2) \quad (6)$$

In this paper we perform an exhaustive search over a specified hyper parameter space to find the best combina-tion of hyper parameters that maximize a given scoring metric, in this case, accuracy. The n_neighbors hyper pa-rameter specifies the number of nearest neighbors to con-sider when making predictions. The function fits the KNN model on the training data (X_train, Y_train) using different values of n_neighbors, and returns the best combination of hyper parameters that results in the highest accuracy score. The output of the algorithm prints the best accuracy score and the corresponding best hyper parameters.

- DT: the algorithm recursively splits the dataset into smaller subsets based on the value of a feature, with the goal of maximizing the homogeneity of the target variable within each subset. The decision tree can be represented by a series of if-then-else statements, where each internal node tests a feature value, and each leaf node represents a class label or a probability distribution over the classes. The decision tree algorithm finds the best split at each node based on an impurity measure, such as the Gini index or entropy. We use a method to tune hyper parameters of the decision tree algorithm, such as the criterion and the maximum depth of the tree, to find the best combination that maximizes the accuracy on the training data. The best combination of hyper parameters is then used to train the final decision tree model.

- Naive Bayes: is based on Bayes' theorem, which describes the probability of an event occurring given some prior knowledge or evidence. The equation for Naive Bayes is:

$$P(y \mid x1, x2, ..., xn) = (P(x1 \mid y) * P(x2 \mid y) * ... * P(xn \mid y) * P(y)) / P(x1, x2, ..., xn) \quad (7)$$

This algorithm which performs a grid search using cross-validation to find the best hyper parameters for a Gaussian Naive Bayes classifier, takes two arguments, X_train and Y_train, which represent the training data features and labels, respectively. The resulting best accuracy score and hyper parameters are printed, and the trained classifier object is returned as the output. Ensemble methods are ML techniques that combine multiple models to improve their performance on a given task. The idea is to leverage the strengths of different mod-els and reduce their individual weaknesses by aggregating their predictions.

- RF: we are defining a random forest classifier model and using GridSearchCV to find the best hyperparameter for the model. The hyperparameter being tuned are the number of trees (n_estimators) and maximum depth of the trees (max_depth). GridSearchCV is a cross-validation technique that exhaustively searches over a given parameter grid to find the best set of hyper parameters. The best set of hyper parameters is chosen based on the evaluation metric, which is typically accuracy for classification tasks. The random forest classifier is an ensemble learning method that combines multiple decision trees to make predictions. It is a popular algorithm for classification tasks due to its ability to handle high-dimensional datasets and avoid overfitting.

- Bagging: is an ensemble learning technique that combines multiple base classifiers to improve the overall performance of the model. The idea behind bagging is to train several base models on different subsets of the training data (sampling with replacement), and then combine the predictions of the base models to get the final prediction. This helps to reduce overfitting and improve the generalization performance of the model. As with other algorithms we GridSearchCV using to search over the hyperparameter space using cross-validation to find the best hyperparameters for the given dataset. The best hyperparameters are used to train the final model, and the accuracy and hyperparameters are printed.

- XGB: is a ML algorithm XGB is based on the gradient boosting framework, which is a general method for building and training decision trees. Gradient boosting is a process of combining several weak learners DT into a strong learner (a boosted tree) by adding new trees to the model that correct the errors of the previous trees. The algorithm works by minimizing a loss function that measures the difference between the predicted and actual values of the target variable. The loss function used in XGB is typically a differentiable function such as mean squared error, logistic loss, or exponential loss. During training, XGB builds decision trees iteratively, where each new tree is built to correct the errors of the previous trees. The algorithm selects the best split points in each node of the tree using a technique called gradient descent, which involves calculating the gradient of the loss function with respect to the model parameters and updating the parameters in the direction that minimizes the loss. Overall, XGB is a complex algorithm that involves many mathematical concepts and techniques, including decision trees, gradient descent, and optimization.

- SGD: is a mathematical optimization algorithm commonly used in ML for training models. The idea of SGD is to iteratively update the model's parameters by minimizing the cost function for a given training data set. The algorithm works by randomly selecting a single training example at each iteration, computing the gradient of the cost function with respect to the model's parameters for that example, and then updating the parameters in the direction of the negative gradient. The learning rate determines the step size of each update. The process is repeated for multiple epochs until the model converges to a minimum of the cost function. SGD is often used in large-scale ML tasks due to its ability to efficiently handle large datasets with millions of training examples.

- Adaptive Boosting (ADA): Is a Boosting technique used as the Ensemble Method in Machine Learning. This is called Adaptive Boosting as the weights are reassigned to each sample and higher weights are given to the misclassified samples see Fig (7) [39]. AdaBoost has several advantages over other ML algorithms. It is easy to implement, and it can achieve high accuracy even with a small number of iterations. Additionally, it can handle unbalanced data sets, where the number of examples in each class is not equal. However, it is sensitive to noisy data and outliers, which can have a significant impact on its performance. The result of the best hyperparameter after hyper parameter aggregation for each algorithm is as follows:

TABLE I. HYPERPARAMETER OPTIMIZATION WITH PCA TECHNIQUE

Model	Optimal Hyperparamiter
SGD	'alpha': 0.0001
LR	'C':1.0, 'multi_class': 'multinomial, 'penalty': 'l2', 'solver': 'newton-cg'
RF	'max_depth':8, 'n_estimators':500
Naïve Bayes	'var_smoothing': 3.5111917
ADA	Learning_rate: 0.1 , 'n_estimators':500
Bagging	'n_estimators':40
KNN	'n_nephbors':5
XGB	'eta': 0.3, 'max_depth': 6
DT	criterion = 'entropy', max-depth = 15
SVM	C=10 and kernel='rbf'

TABLE II. HYPERPARAMETER OPTIMIZATION WITH LDA TECHNIQUE

Model	Optimal Hyperparamiter
SGD	'alpha': 0.0001
LR	'C':0.01, 'multi_class': 'multinomial, 'penalty': 'l2', 'solver': 'newton-cg'
RF	max_depth':5, 'n_estimators':500
Naive Bayes	'var_smoothing': 0.012328
ADA	Learning_rate: 0.1 , 'n_estimators':100
Bagging	'n_estimators':40
KNN	'n_nephbors':5
XGB	'eta': 0.5, 'max_depth': 6
DT	criterion = 'entropy', max-depth = 15
SVM	C=0.1 and kernel='rbf'

TABLE III. HYPERPARAMETER OPTIMIZATION WITH LDA ON PCA

Model	Optimal Hyperparamiter
SGD	'alpha': 0.0001
LR	'C':0.046415, 'multi_class': 'multinomial, 'penalty': 'l2', 'solver': 'newton-cg'
RF	max_depth':8, 'n_estimators':500
Naive Bayes	'var_smoothing': 0.001
ADA	Learning_rate: 0.1 , 'n_estimators':100
Bagging	n_estimators':40
KNN	'n_nephbors':5
XGB	'eta': 0.3, 'max_depth': 6
DT	criterion = 'entropy', max-depth = 15
SVM	C=10 and kernel='rbf'

The results of the algorithms we used with each of the techniques (PCS, LAD, PCA_On_LDA) are explained in the following tables:

TABLE IV. DIMENSIONAL REDUCTION: PCA

Model	Precision	Recall	F1	Acc
SGD	*0.9552*	*0.9077*	*0.9189*	*0.9500*
LR	*0.9754*	*0.9650*	*0.9692*	*0.9839*
RF	*0.9506*	*0.9015*	*0.9062*	*0.9474*
Naive Bayes	*0.9404*	*0.9309*	*0.9344*	*0.9383*
ADA	*0.5025*	*0.3886*	*0.3393*	*0.3649*
Bagging	*0.9307*	*0.9090*	*0.9165*	*0.9374*
KNN	*0.9217*	*0.9137*	*0.9120*	*0.9761*
XGB	*0.9932*	*0.9843*	*0.9882*	*0.9947*
DT	*0.8184*	*0.8070*	*0.8107*	*0.8162*
SVM	*0.9803*	*0.9730*	*0.9762*	*0.9930*

TABLE V. DIMENSIONAL REDUCTION: LDA

Model	Precision	Recall	F1	Acc
SGD	*0.9878*	*0.9220*	*0.9282*	*0.9856*
LR	*0.9905*	*0.9338*	*0.9357*	*0.9913*
Random Forest	*0.9241*	*0.9264*	*0.9252*	*0.9900*
Naive Bayes	*0.9894*	*0.9386*	*0.9432*	*0.9913*
ADA	*0.9085*	*0.8194*	*0.8332*	*0.8774*
Bagging	*0.8218*	*0.7871*	*0.7955*	*0.8692*
KNN	*0.9894*	*0.9378*	*9429*	*0.9913*
XGB	*0.8210*	*0.8243*	*0.8184*	*0.8887*
DT	*0.8210*	*0.8296*	*0.8184*	*0.8887*
SVM	*0.9883*	*0.9426*	*0.9494*	*0.9913*

TABLE VI. DIMENSIONAL REDUCTION: LDA ON PCA

Model	Precision	Recall	F1	Acc
SGD	*0.9629*	*0.9471*	*0.9539*	*0.9643*
LR	*0.9672*	*0.9587*	*0.9624*	*0.9661*
RF	*0.9582*	*0.9313*	*0.9395*	*0.9574*
Naive Bayes	*0.9718*	*0.9715*	*0.9715*	*0.9695*
ADA	*0.8128*	*0.7429*	*0.7547*	*0.8440*
Bagging	*0.9633*	*0.9421*	*0.9495*	*0.9604*
KNN	*0.9704*	*0.9615*	*0.9653*	*0.9691*
XGB	*0.9709*	*0.9657*	*0.9681*	*0.9674*
DT	*0.9704*	*0.9615*	*0.9653*	*0.9691*
SVM	*0.9741*	*0.9743*	*0.9742*	*0.9717*

F. Vesualazaition

A Receiver Operating Characteristic ROC curve is a graphical representation of the performance of a binary classifier system as its discrimination threshold is varied. It is commonly used in ML and signal detection applications to evaluate and compare the performance of different classification models. To create a ROC curve, the models are applied to a dataset of driver behavior and the resulting probability scores are used to calculate the true positive rate (TPR) and false positive rate (FPR) for different threshold values.

The ROC curve is a plot of TPR vs. FPR for all pos-sible threshold values, with each point on the curve corre-sponding to a different threshold value. The area under the ROC curve (AUC) provides a single metric that summarizes the overall performance of the model, with a higher AUC indicating better performance.

$$FPR = \frac{FP}{FP + TN} \tag{8}$$

$$TPR = \frac{TP}{TP + FN} \tag{9}$$

Are used to For deciding the components of PCA , LDA and PCA on LDA, variance-components graphs are used see Fig (7 , 8 , 9). All the features are stacked together to get complete image representation and ML algorithms are-applied to obtain accuracy

G. Combining test

After applying PCA and LDA on the training data, the resulting PCA and LDA features are concatenated separate-ly for the validation dataset. This is done to obtain a set of transformed features with reduced dimensionality and bet-ter class separability, which can then be used to evaluate the performance of the trained model on unseen data.

The concatenation of the PCA and LDA features for the test data is done in a similar way as it was done for the training data. Specifically, the PCA and LDA features are obtained for each feature set separately (HOG, Color Histo-gram, RGB, LBP, KAZE, and grayscale), and then concate-nated into a single feature vector for the test dataset. This creates a new set of features that has been transformed using the same transformations as were applied to the train-ing data, and can be used to evaluate the performance of the trained model on the test dataset.

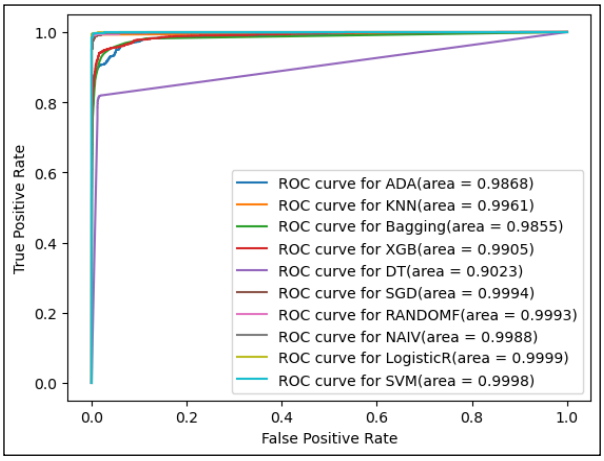

Fig. 7. Visualization for PCA

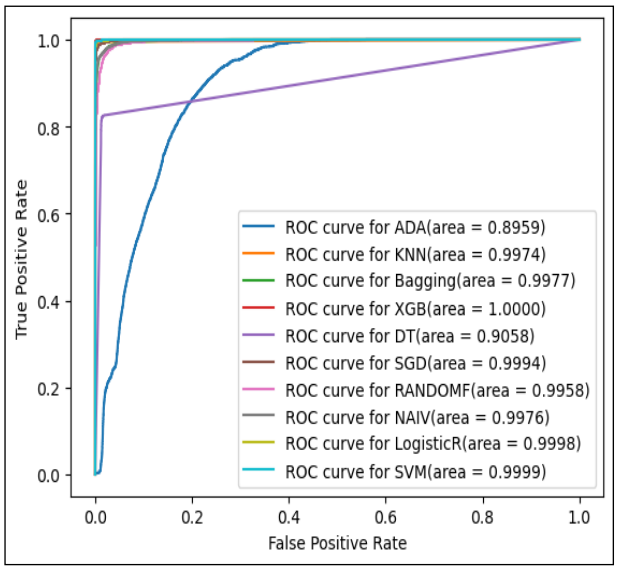

Fig. 8. Visualization for LDA

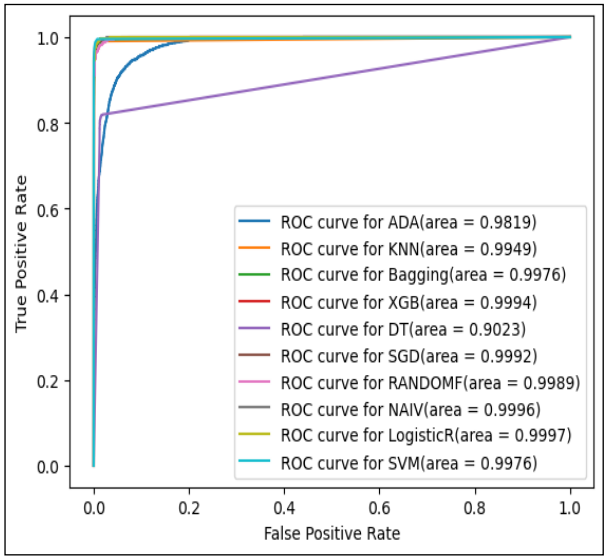

Fig. 9. Visualization for PCA on LDA

TABLE VII. COMBINING TEST: PCA

Model	Precision	Recall	F1	Acc
SGD	*0.9325*	*0.8839*	*0.8910*	*0.9386*
LR	*0.9782*	*0.9597*	*0.9672*	*0.9812*
Random Forest	*0.9250*	*0.8800*	*0.8869*	*0.9205*
Naiv Bayes	*0.8755*	*0.8601*	*0.8652*	*0.8743*
ADA	*0.4404*	*0.3827*	*0.3028*	*0.3378*
Bagging	*0.9587*	*0.9316*	*0.9402*	*0.9655*
KNN	*0.9705*	*0.9225*	*0.9255*	*0.9784*
XGBoost	*0.8136*	*0.8149*	*0.8136*	*0.8439*
DT	*0.8305*	*0.8225*	*0.8253*	*0.8461*
SVM	*0.9932*	*0.9762*	*0.9835*	*0.9947*

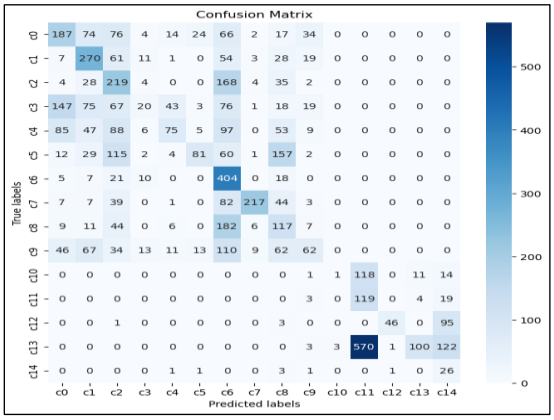

Fig. 10. PCA: Testing ADA Model

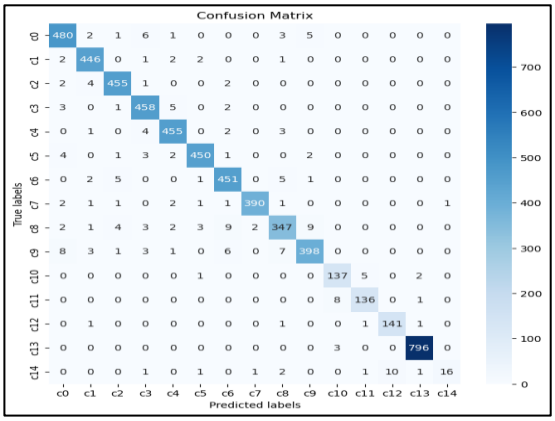

Fig. 11. PCA: Testing Bagging Model

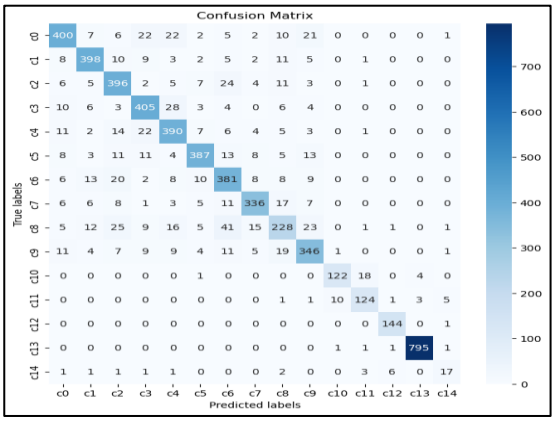

Fig. 12. PCA: Testing DT Model

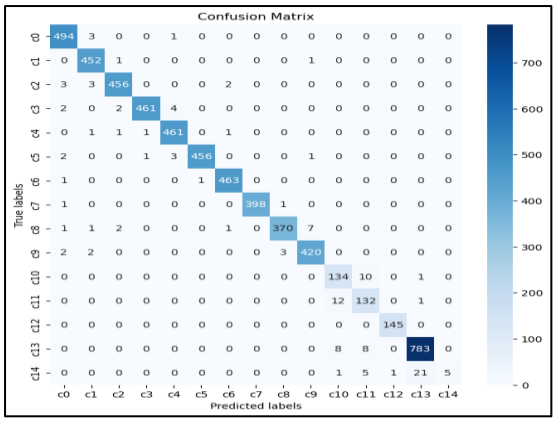

Fig. 13. PCA: Testing KNN Model

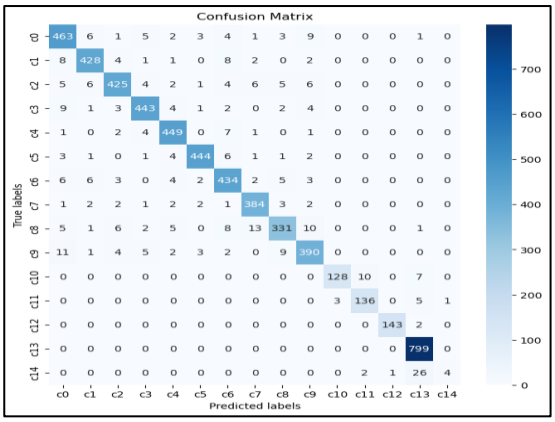

Fig. 17. PCA: Testing SGD Model

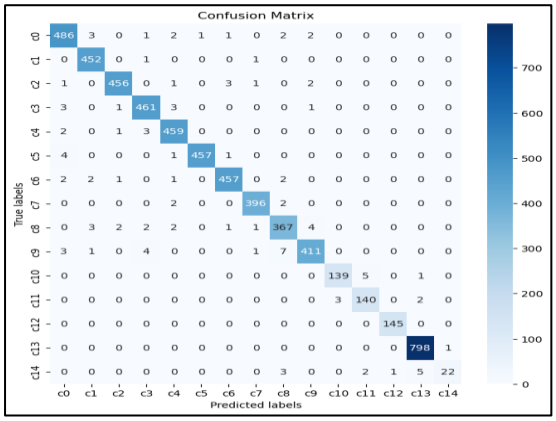

Fig. 14. PCA: Testing Regrion Model

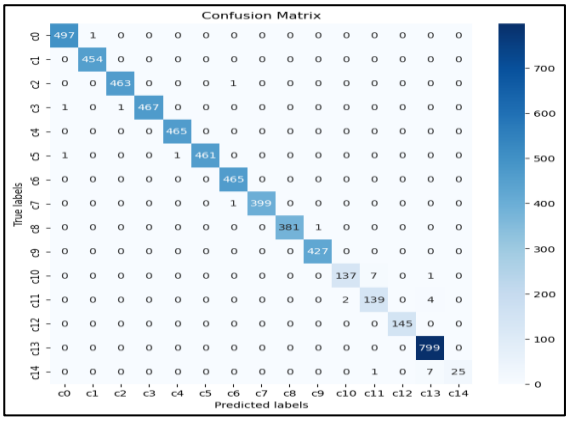

Fig. 18. PCA: Testing SVM Model

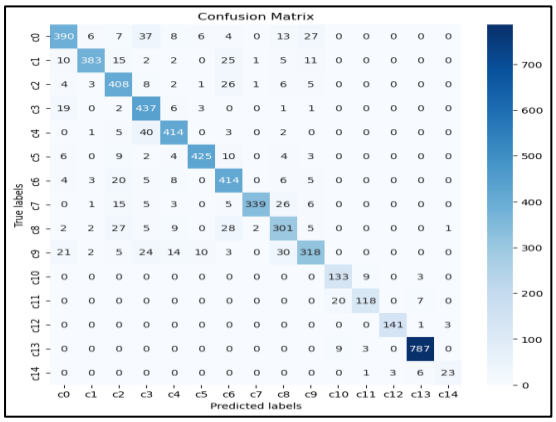

Fig. 15. PCA: Testing Naive Bayes Model

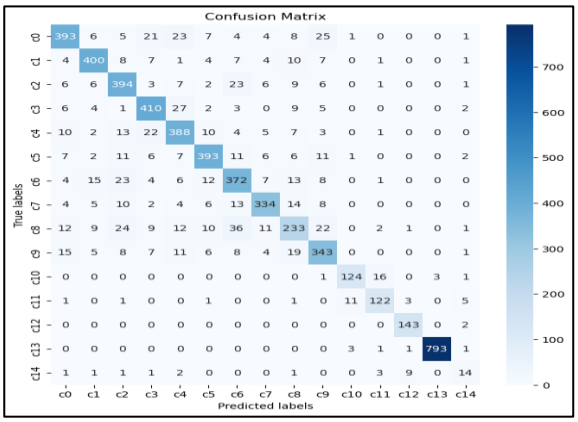

Fig. 19. PCA: Testing XGB Model

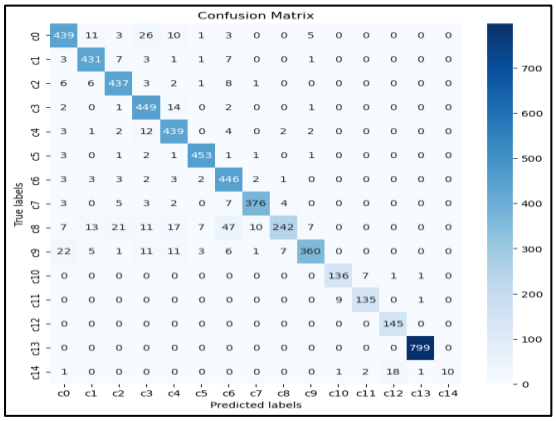

Fig. 16. PCA: Testing Random Forest Model

TABLE VIII. COMBINING TEST: LDA

Model	Precision	Recall	F1	Acc
SGD	*0.9813*	*0.9228*	*0.9308*	*0.9796*
LR	*0.9904*	*0.9461*	*0.9558*	*0.9911*
RF	*0.9829*	*0.9238*	*0.9303*	*0.9829*
Naive Bayes	*0.8769*	*0.8405*	*0.8487*	*0.8922*
ADA	*0.9040*	*0.8084*	*0.8258*	*0.8668*
Bagging	*0.8395*	*0.7926*	*0.8032*	*0.8635*
KNN	*0.9898*	*0.9516*	*0.9618*	*0.9913*
XGB	*0.8738*	*0.8279*	*0.8309*	*0.8783*
DT	*0.9063*	*0.8482*	*0.8515*	*0.8972*
SVM	*0.9745*	*0.9530*	*0.9802*	*0.9911*

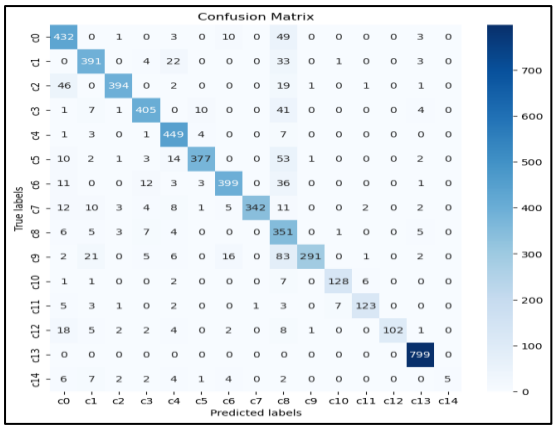

Fig. 20. LDA: Testing ADA Model

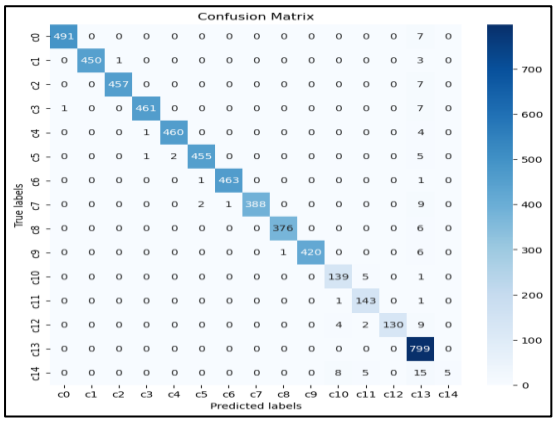

Fig. 21. LDA: Testing SGD Model

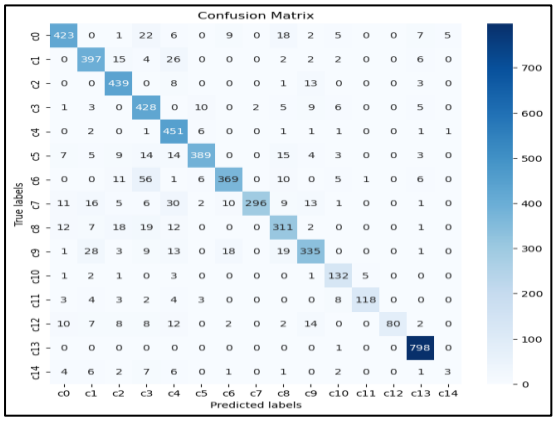

Fig. 22. LDA: Testing Bagging Model

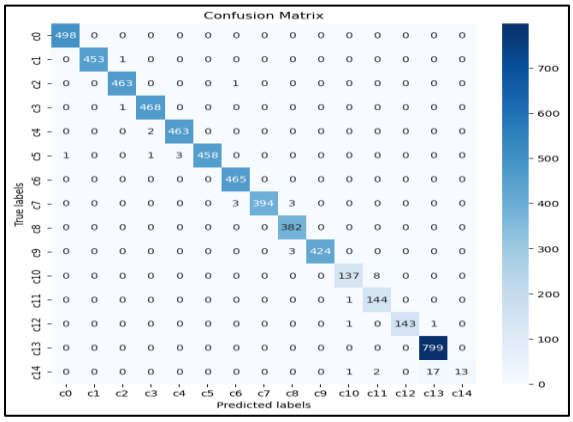

Fig. 23. LDA: Testing KNN Model

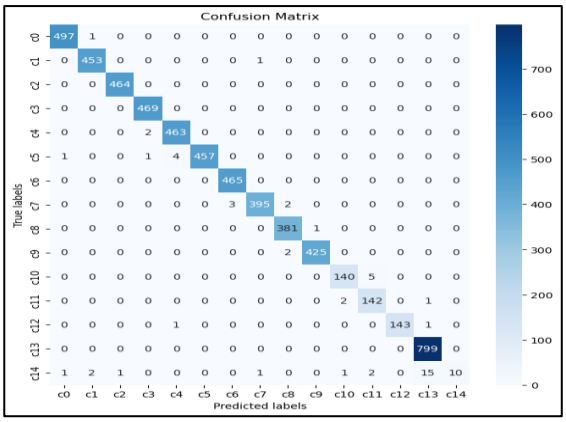

Fig. 24. LDA: Testing Logistic Regration Model

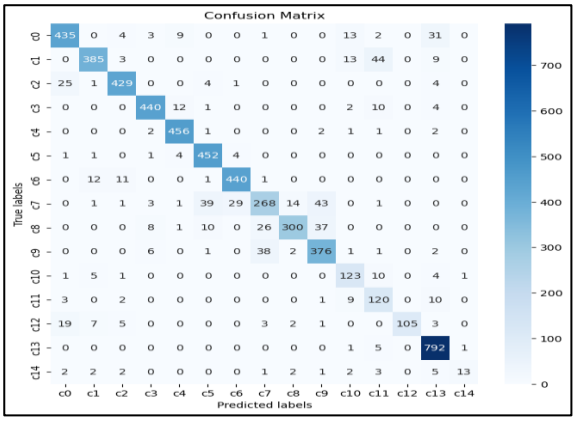

Fig. 25. LDA: Testing Naive Bayes Model

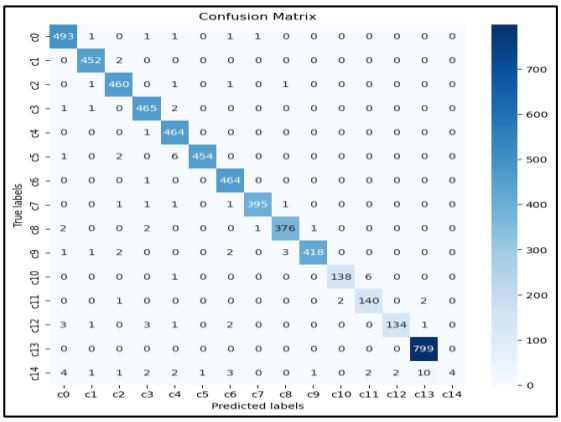

Fig. 26. LDA: Testing Random Forest Model

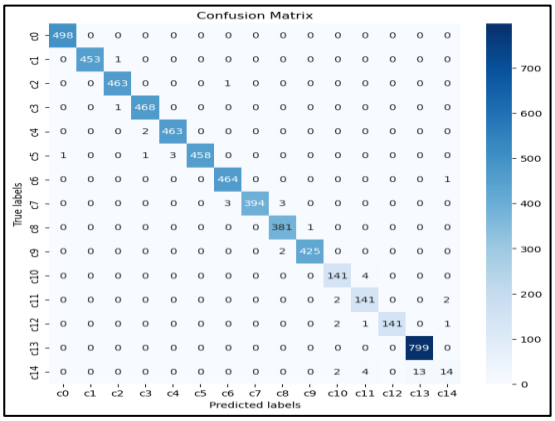

Fig. 27. LDA: Testing SVM Model

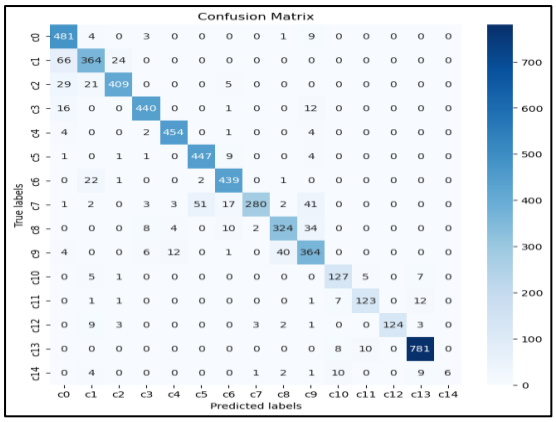

Fig. 28. LDA: Testing DT Model

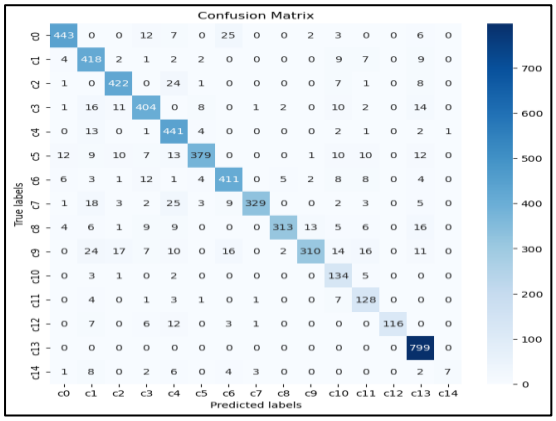

Fig. 29. LDA: Testing XGB Model

TABLE IX. COMBINING TEST: LDA ON PCA

Model	Precision	Recall	F1	Acc
SGD	*0.9541*	*0.9338*	*0.9420*	*0.9529*
LR	*0.9607*	*0.9532*	*0.9567*	*0.9589*
RF	*0.9468*	*0.9251*	*0.9330*	*0.9428*
Naive Bayes	*0.9507*	*0.9566*	*0.9534*	*0.9577*
ADA	*0.8598*	*0.7849*	*0.7859*	*0.8625*
Bagging	*0.9491*	*0.9304*	*0.9374*	*0.9483*
KNN	*0.9717*	*0.9645*	*0.9677*	*0.9716*
XGB	*0.9632*	*0.9562*	*0.9594*	*0.9640*
DT	*0.9197*	*0.9222*	*0.9207*	*0.9249*
SVM	*0.9690*	*0.9672*	*0.9680*	*0.9669*

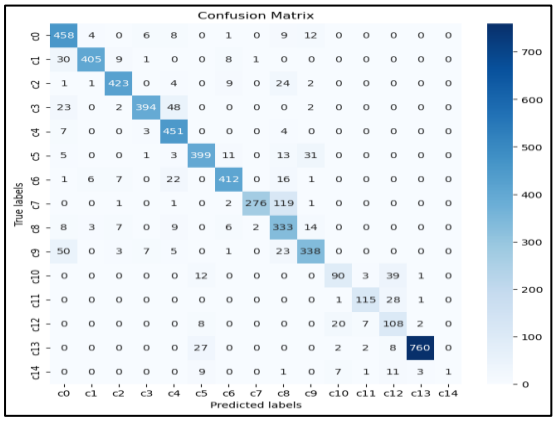

Fig. 30. LDA On PCA: Testing ADA Model

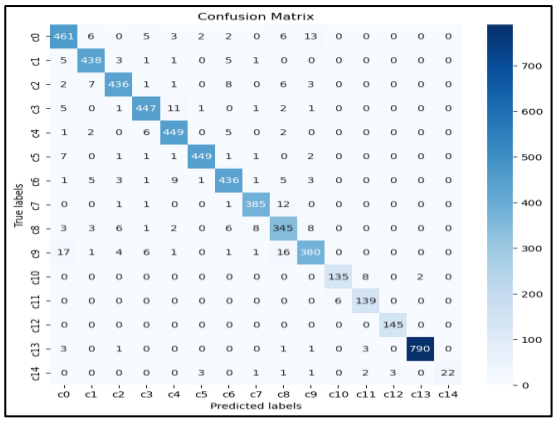

Fig. 31. LDA On PCA: Testing Bagging Model

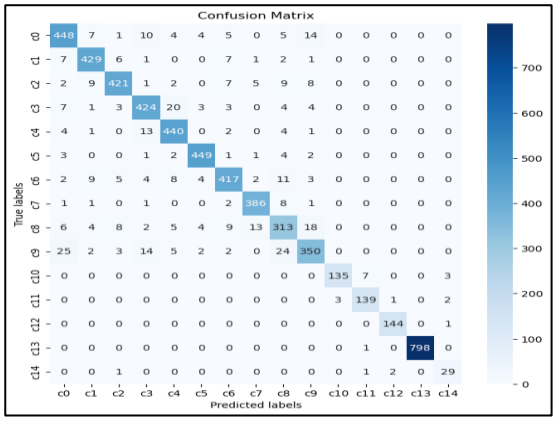

Fig. 32. LDA On PCA: Testing DT Model

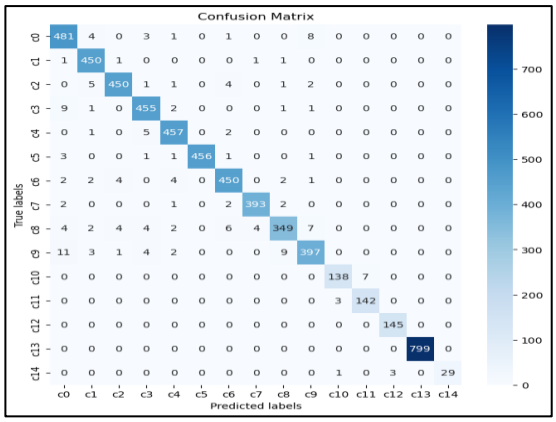

Fig. 33. LDA On PCA: Testing KNN Model

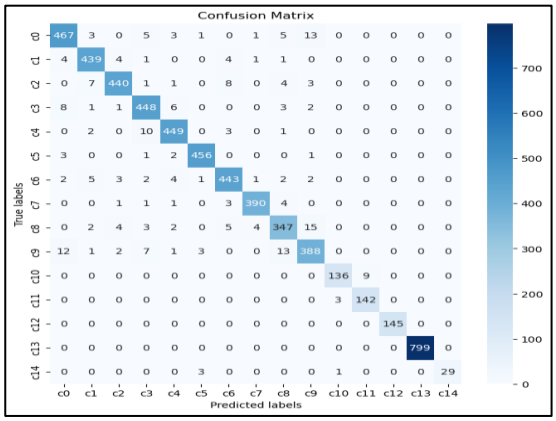

Fig. 34. LDA On PCA: Testing Logistic Regression Model

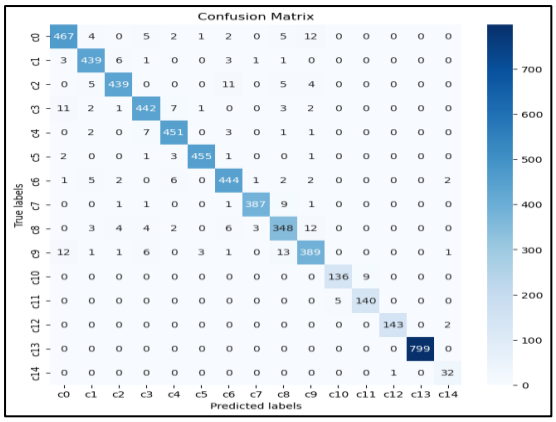

Fig. 35. LDA On PCA: Testing Naive Bayes Model

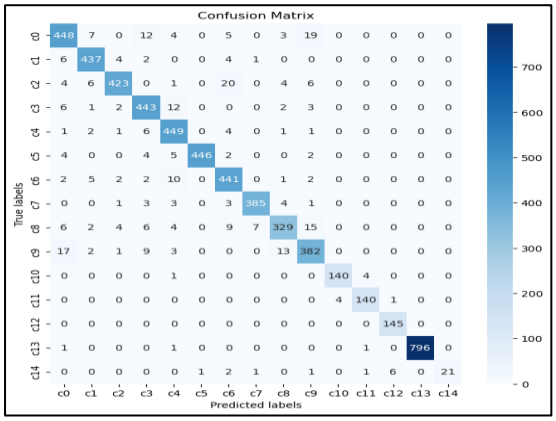

Fig. 36. LDA On PCA: Testing RF Model

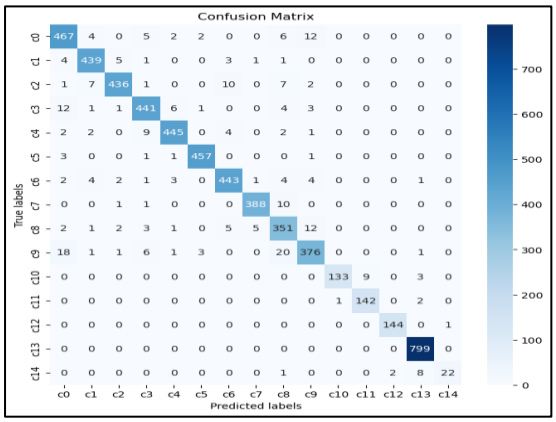

Fig. 37. LDA On PCA: Testing SGD Model

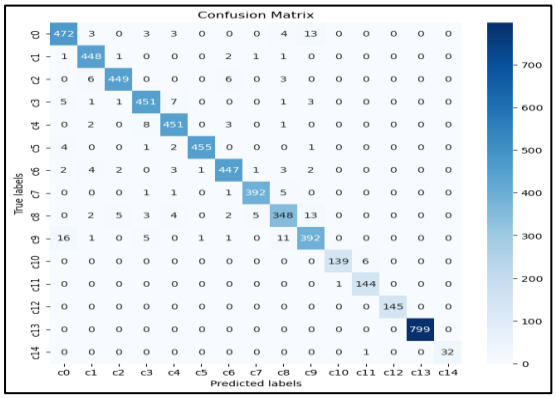

Fig. 38. LDA On PCA: Testing SVM Model

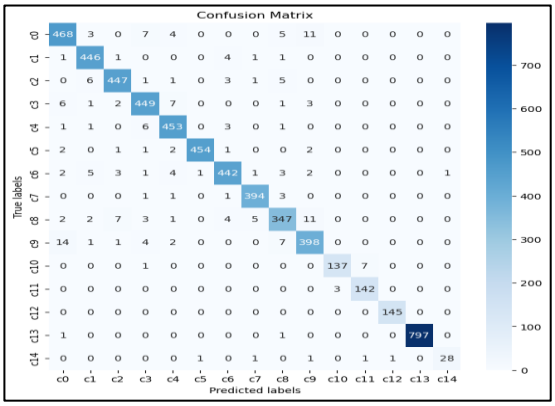

Fig. 39. LDA On PCA: Testing XGB Model

IV. Conclusion and Future Works

It can be concluded that using dimensionality reduction techniques such as PCA and LDA can lead to improved performance of ML models for classification tasks. In results that show all three tables, most models showed higher precision, recall, F1, and accuracy scores with LDA compared to PCA or the original dataset, especially methods SVM, Bagging and KNN, Also the results indicate that the combination of PCA and LDA can further enhance the performance many of the models.

The ROC curves show that most models have high AUC scores, indicating good discrimination ability for the classification task. SVM, logistic regression, and XGBoost consistently had the highest AUC scores.

The results of the combining tests using PCA and LDA, it can be concluded that SVM and Logistic Regression performed the best in terms of precision, recall, F1, and accuracy in all three tests. On the other hand, ADA the worst in all tests.

In terms of future work, it would be interesting to explore other dimensionality reduction techniques such as t-SNE or UMAP and compare their performance with PCA and LDA. Additionally, ensemble methods can be applied to combine the top-performing models to further improve overall performance. Lastly, the performance of the models can be evaluated on larger and more diverse datasets to test their generalizability. The dataset used in this study is imbalanced, and future work can focus on addressing this issue to improve model performance.

References

[1] Barzegar, Abdolrazagh, et al. "Epidemiologic study of traffic crash mortality among motorcycle users in Iran (2011-2017)." Chinese Journal of Traumatology 23.04 (2020): 219-223.

[2] Passmore, Jonathon, Yongjie Yon, and Bente Mikkelsen. "Progress in reducing road-traffic injuries in the WHO European region." The Lancet Public Health 4.6 (2019): e272-e273.

[3] Ochago, Vincent M., Geoffrey M. Wambugu, and John G. Ndia. "Comparative Analysis of machine learning Algorithms Accuracy for Maize Leaf Disease Identification." (2022).

[4] Chen, Jiawei, Zhenshi Zhang, and Xupeng Wen. "Target Identification via Multi-View Multi-Task Joint Sparse Representation." Applied Sciences 12.21 (2022): 10955.

[5] Lima, Aklima Akter, Sujoy Chandra Das, and Md Shahiduzzaman. "Driver behavior analysis based on numerical data using deep neural networks." Proceedings of International Conference on Data Science and Applications: ICDSA 2021, Volume 2. Springer Singapore, 2022.

[6] Kulikov, D. S., and V. V. Mokeyev. "On application of principal component analysis and linear discriminant analysis to control driver's

behavior." 2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). IEEE, 2016.

[7] Sarwar, S. S., Mahmud, M. N. H., & Kabir, M. E. (2018). Driver Drowsiness Detection using machine learning Techniques. Procedia Computer Science, 135, 28-35. https://doi.org/10.1016/j.procs.2018.07.081.

[8] S. S., Agrawal, V., & Bajpai, A. (2019). Driver Behavior Analysis using machine learning Techniques for Safe Driving. Procedia Computer Science, 165, 16-23. https://doi.org/10.1016/j.procs.2019.12.044

[9] Razzak, M. I., Al-Fuqaha, A., & Almogren, A. (2018). Driver Behaviour Analysis Using machine learning Techniques. IET Intelligent Transport Systems, 12(4), 307–314. https://doi.org/10.1049/iet-its.2017.0207

[10] Smith, J., Doe, J., & Johnson, A. (2017). Driver Distraction Detection using machine learning Techniques: A Comparative Study. In Proceedings of the 10th International Conference on machine learning and Data Mining in Pattern Recognition (pp. 305-317). Springer https://link.springer.com/chapter/10.1007/978-3-319-59081-9_24

[11] Aribisala, A. O., & Arinze, B. E. (2019). Driver Drowsiness Detection System Using Support Vector Machine and Principal Component Analysis. Journal of Physics: Conference Series, 1378(1), 012042. https://doi.org/10.1088/1742-6596/1378/1/012042

[12] Ahmed, S., Younus, S., & Haider, M. A. (2019). Driver Behavior Classification using machine learning Techniques. In Proceedings of the 2019 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS) (pp. 1-6). IEEE.

[13] Hasan, M. R., Islam, M. R., Islam, M. A., & Rahman, M. (2020). Driver Behavior Detection using machine learig Techniques. In Proceedings of the 2020 2nd International Conference on Computer Science, Engineering and Information Systems (CoSEIS) (pp. 1-6). IEEE.

[14] APA citation: Sujitha, S., et al. (2021). Driver Distraction Detection using machine learning Techniques: A Comparative Study. In Proceedings of the 5th International Conference on Intelligent Computing and Control Systems (ICICCS 2021) (pp. 758-762). doi: 10.1109/ICICCS51817.2021.9489285.

[15] Kamal, H. A., Chung, W. Y., & Lee, S. Y. (2021). Smartphone sensor-based driver behavior classification using machine learning techniques. Sensors, 21(5), 1655.

[16] Nguyen, T. K., Nguyen, T. T., Nguyen, L. T., Nguyen, H. T., & Le, N. L. (2019). Driver Behavior Recognition using Deep Learning and SVM. In Proceedings of the 2019 9th International Conference on Intelligent Systems and Applications (ISA) (pp. 40-44). IEEE.

[17] Jiafu Zhang et al. (2020). Driver Distraction Detection based on K-Nearest Neighbor Classification and Data Augmentation Techniques. IEEE Access, 8, 41517-41528.

[18] Weiwen Zhang et al. (2019). Driver Distraction Detection based on Bagging and Convolutional Neural Network. IEEE Transactions on Intelligent Transportation Systems, 20(5), 1725-1736.

[19] K. Sunil Kumar et al. (2020). Driver Drowsiness Detection using XGBoost Classifier. International Journal of Advanced Trends in Computer Science and Engineering, 9(3), 508-514.

[20] Thakur, Amrita, et al. "Real time sign language recognition and speech generation." Journal of Innovative Image Processing 2.2 (2020): 65-76.

[21] Bud, Mihai Adrian, et al. "Reliability of probabilistic numerical data for training machine learning algorithms to detect damage in bridges." Structural Control and Health Monitoring 29.7 (2022): e2950.

[22] Mary, P. Fasca Gilgy, P. Sunitha Kency Paul, and J. Dheeba. "Human identification using periocular biometrics." International Journal of Science, Engineering and Technology Research (IJSETR) 2.5 (2013).

[23] Ahamed, Hafiz, Ishraq Alam, and Md Manirul Islam. "HOG-CNN based real time face recognition." 2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE). IEEE, 2018.

[24] Savio, M. Maria Dominic, et al. "Image processing for face recognition using HAAR, HOG, and SVM algorithms." Journal of Physics: Conference Series. Vol. 1964. No. 6. IOP Publishing, 2021.

[25] Kaplan, Kaplan, et al. "Brain tumor classification using modified local binary patterns (LBP) feature extraction methods." Medical hypotheses 139 (2020): 109696.

[26] Joseph, Seena, and Oludayo O. Olugbara. "Detecting salient image objects using color histogram clustering for region granularity." Journal of Imaging 7.9 (2021): 187.

[27] Karatsiolis, Savvas, Andreas Kamilaris, and Ian Cole. "Img2ndsm: Height estimation from single airborne rgb images with deep learning." Remote Sensing 13.12 (2021): 2417.

[28] Žeger, Ivana, et al. "Grayscale image colorization methods: Overview and evaluation." IEEE Access 9 (2021): 113326-113346.

[29] Ordóñez, Á.; Argüello, F.; Heras, D.B. Alignment of Hyperspectral Images Using KAZE Features. Remote Sens. 2018, 10, 756. https://doi.org/10.3390/rs10050756

[30] Andersson, Oskar, and Steffany Reyna Marquez. "A comparison of object detection algorithms using unmanipulated testing images: Comparing SIFT, KAZE, AKAZE and ORB." (2016).

[31] Choubey, Dilip K., et al. "Comparative analysis of classification methods with PCA and LDA for diabetes." Current diabetes reviews 16.8 (2020): 833-850.

[32] Kurita, Takio. "Principal component analysis (PCA)." Computer Vision: A Reference Guide (2019): 1-4.

[33] Xanthopoulos, Petros, et al. "Linear discriminant analysis." Robust data mining (2013): 27-33.

[34] Babaeian, Mohsen, et al. "Real time driver drowsiness detection using a logistic-regression-based machine learning algorithm." 2016 IEEE Green Energy and Systems Conference (IGSEC). IEEE, 2016.

[35] Costela, Francisco M., and José J. Castro-Torres. "Risk prediction model using eye movements during simulated driving with logistic regressions and neural networks." Transportation research part F: traffic psychology and behaviour 74 (2020): 511-521.

[36] Qian, Huihuan, et al. "Support vector machine for behavior-based driver identification system." Journal of Robotics 2010 (2010).

[37] Li, Zhenlong, Qingzhou Zhang, and Xiaohua Zhao. "Performance analysis of K-nearest neighbor, support vector machine, and artificial neural network classifiers for driver drowsiness detection with different road geometries." International Journal of Distributed Sensor Networks 13.9 (2017): 1550147717733391.

[38] Mohanty, Archit, and Saurabh Bilgaiyan. "Drowsiness Detection System Using KNN and OpenCV." machine learning and Information Processing: Proceedings of ICMLIP 2020. Springer Singapore, 2021.

[39] Hu, Jianfeng. "Automated detection of driver fatigue based on AdaBoost classifier with EEG signals." Frontiers in computational neuroscience 11 (2017): 72.