Distracted Driver Detection System

A SEMINAR REPORT

Submitted in partial fulfillment of the

requirements for the award for the degree of

BACHELOR OF TECHNOLOGY

In ELECTRICAL ENGINEERING

By

VIBHANSHU BOTKE

(ROLL NO: U21EE071)

Under the supervision of

Dr. RAKESH MAURYA

DEPARTMENT OF ELECTRICAL ENGINEERING

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY (SVNIT)

SURAT, GUJARAT-395007(INDIA)

OCTOBER 2023

सरदार वल्लभभाई राष्ट्रीय प्रौद्योगिकी संस्थान, सूरत
Sardar Vallabhbhai National Institute of Technology
(SVNIT), Surat, Gujarat - 395007

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in this seminar report entitled **"Distracted Driver Detection System"** in partial fulfillment of the requirements for the award of the BACHELOR OF TECHNOLOGY in ELECTRICAL ENGINEERING and submitted in The Department of Electrical Engineering of the Sardar Vallabhbhai National Institute of Technology(SVNIT), Surat, Gujarat is an authentic record of my own work carried by me under the guidance of **Dr. Rakesh Maurya**, EED, SVNIT

The matter submitted in this report has not been submitted by me or anyone for the award of any other degree or in any other institute.

Vibhanshu Botke

This is to certify that the above statement made by the student is correct to the best of our knowledge.

Date: 20/10/2023 Place: SVNIT, Surat

Dr. Rakesh Maurya

Supervisor

सरदार वल्लभभाई राष्ट्रीय प्रौद्योगिकी संस्थान, सूरत
Sardar Vallabhbhai National Institute of Technology
(SVNIT), Surat, Gujarat - 395007

Examiner's Approval Certificate

The seminar entitled "**Distracted Driver Detection System**" submitted by **Vibhanshu Botke (U21EE071)** in the partial fulfillment of the requirements for the award of the degree of "Bachelor of Technology in Electrical Engineering" of the Sardar Vallabhbhai National Institute of Technology, Surat is hereby approved

Date: 20/10/2023

Place: SVNIT, Surat

(**Dr. Rakesh Maurya**)

Supervisor

Examiner's Signature

Examiner's Signature

Chairperson

ACKNOWLEDGEMENT

For the success of this seminar, a lot of guidance and assistance was required from experienced faculties apart from my own efforts and I feel extremely privileged to get all this along the course of this seminar.

I take this opportunity to express my sincere gratitude to my guide Dr. Rakesh Maurya, without whom the seminar would not have been successfully completed. His encouragement proved very useful to me at every stage. He guided me in the best way possible and motivated me to perform exceptionally in this seminar. His guidance paved the way for the successful completion of this seminar and I heartily thank him for his exemplary guidance, constant encouragement, and careful monitoring throughout the seminar.

Finally, I am thankful to everyone who helped me in this seminar and were a constant support to me during this period.

ABSTRACT

Driving a car is a difficult task that needs driver full attention. Any activity that diverts the driver's attention from the road constitutes distracted driving. The evidence is mounting that driver distraction is a major factor in car accidents and other incidents. The increased use of so-called in-vehicle information systems (IVIS) and partially autonomous driving assistance systems (PADAS) has drawn attention in particular to growing safety concerns. In order to properly modify IVIS and PADAS and prevent or lessen any potential negative effects, it is essential to identify the driver's state.

By using various Machine Learning Models to categorize the provided images to different categories of Distraction, this project aims to determine whether the car driver is operating the vehicle safely or engaging in any activity that could cause an accident or any harm to others. This research can be further extended by contrasting various Machine Learning models to ascertain the accuracy based on individual models.

CONTENTS

1. INTRODUCTION

The rise in traffic fatalities and accidents in recent years has raised serious concerns, which is largely because distracted driving is so common. Three main categories of distraction have been established by numerous studies: visual (driver's eyes off the road), manual (driver's hands off the wheel), and cognitive (driver's mind off the task of driving).

The National Highway Traffic Safety Administration (NHTSA) reports that 36,750 people died in car accidents in 2018, with distracted driving being to blame for 12% of those fatalities. Texting is the most hazardous distraction. To send or read a text, you must take your eyes off the road for five seconds. That is the same as moving at 55 mph while closing your eyes across a football field.

Distraction or inattention among drivers is not universally understood. The linked concepts are frequently discussed in the literature, despite the fact that their definitions and relationships are frequently inconsistent. Furthermore, it's not entirely clear how much driver distraction contributes to collisions. Therefore, according to Wang et al., distraction was a factor in 13.3% of crashes, and "looked but did not see" crashes accounted for 9.7% of crashes. If drowsiness is taken into account, this percentage may even rise (+2.6%). In fact, the majority of traffic accidents (certainly > 80%) are caused by human error or (wrong) human behavior, with mounting evidence pointing to driver distraction and inattention as significant contributing factors in automobile and truck crash incidents.

Despite the lack of agreement among scientists regarding its definition and actual effects, one thing appears to be universal. Driver inattention and distraction pose a serious threat to public safety. Allowing drivers to use these IVIS and PADAS in this situation without compromising safety is a serious problem. One promising approach to solving this issue aims to adapt IVIS technologies to lessen the effects of distraction and PADAS strategies to reduce the effects of distraction on the driving task. This tactic entails categorizing the driver's current state, in this case, a distracted driver.

Machine learning (ML) may offer the appropriate approaches for dealing with such a challenge. ML is a technique for searching through enormous amounts of data for previously unidentified patterns. This method can be applied, for instance, to develop a discriminating model that captures the behavioral differences between drivers who are paying attention and those who are not. The main goal of this paper is to provide a non-intrusive method for a real-time system to recognize and classify driver distraction using ML algorithms (comparison of various approaches) and only vehicle dynamic data as model inputs. Here, we emphasize the visual distraction of the driver, which has been found to be a significant contributing factor in the maneuvers under investigation. Looking away for a brief period of time (at least 1.8 seconds) might be seen in this context as a visual distraction for the driver from their primary task.

1.1 INTRODUCTION TO MACHINE LEARNING

Machine learning (ML) is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. Building algorithms that can take input data and apply statistical analysis to predict an output while updating results as new data becomes available is the fundamental idea behind machine learning. Starting with feeding our machines high-quality data, train them by creating machine learning models using the data and various algorithms. The type of data we have and the type of task we're trying to automate will influence the algorithms we choose.

Methods of Machine Learning

There are three types of machine learning algorithms: -

Supervised Learning

With supervised learning, the program is given labeled input data and the desired results. When given data that has never been seen before, the model is trained until it can recognize the underlying patterns and relationships and produce accurate results.

Types of Supervised Learning

1. Classification - Classification algorithms are used to predict or classify items into discrete valued groups.

2. Regression - Regression tasks are different, as they expect the model to predict continuous data.

Unsupervised Learning

Unsupervised learning is a type of learning where a computer picks up information without any human intervention. The machine is trained using a set of unlabeled, unclassified, or uncategorized data, and the algorithm is required to act independently on that data. Unsupervised learning's objective is to reorganize the input data into fresh features or a collection of objects with related patterns.

Reinforcement Learning

A learning agent in a reinforcement learning method receives a reward for each correct action and receives a penalty for each incorrect action. With the help of these feedbacks, the agent automatically learns and performs better. The agent explores and engages with the environment during reinforcement learning. An agent performs better because its objective is to accrue the most reward points.

1.2 INTRODUCTION TO DEEP LEARNING

Machine learning is simply a neural network with multiple layers and is a subset of deep learning. Additional hidden layers can help to tune and refine for accuracy even if a neural network with only one layer can still make approximation predictions. Through a combination of data inputs, weights, and bias, these elements work together to accurately recognize, classify, and describe objects within the data.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible* layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

In order to properly train the model, backpropagation employs algorithms to quantify prediction errors before changing the weights and biases of the function by going backward through the layers. A neural network may create predictions and correct any faults using forward propagation and backpropagation together. The algorithm continuously improves in accuracy over time.

Convolutional neural networks (CNNs), which are mostly employed in computer vision and image classification applications, are able to recognize patterns and features in an image, enabling tasks like object recognition or detection.

As it makes use of sequential or time series data, recurrent neural networks (RNNs) are frequently utilized in applications like speech and natural language recognition. They use feedback loops to digest a sequence of data that informs the final output. These feedback loops allow for the persistence of information. Oftentimes, this phenomenon is referred to as memory.

1.3 PROPOSED SOLUTION

In the context of a distracted driver detection system powered by machine learning models, the incorporation of cutting-edge driver assistance systems becomes essential. These systems' ability to detect driver distraction has numerous benefits for both enhancing the driving experience and enhancing road safety.

For instance, the intervention of a forward-collision assistance system may be triggered based on the driver's state as identified by the machine learning model. The collision assistance system's function strategies can be modified if distraction is found. Altering the modulation of the brakes or sending out warning signals well in advance of a potential collision are both options. This dynamic response to the distraction level of the driver significantly lowers the likelihood of accidents brought on by inattention, ultimately enhancing road safety.

On the other hand, if the system determines that the driver is not distracted but, for example, intends to pass another vehicle, it can intelligently delay or suppress warning messages, even when approaching a car ahead. This level of contextual awareness, powered by the machine learning model, ensures that drivers are not inundated with pointless alarms during routine driving situations, making driving more pleasant and less unpleasant.

The addition of such intelligent assistance, which understands the driver's intention and state, not only increases safety margins but also boosts user acceptance of the system as a whole. The Distracted Driver Detection System lessens the risks related to distraction by lowering false alarms and offering assistance specific to the driver's situation.

2. LITERATURE SURVEY

The review of some of the pertinent and important work from the literature for detecting distracted driving is summarized in this part.

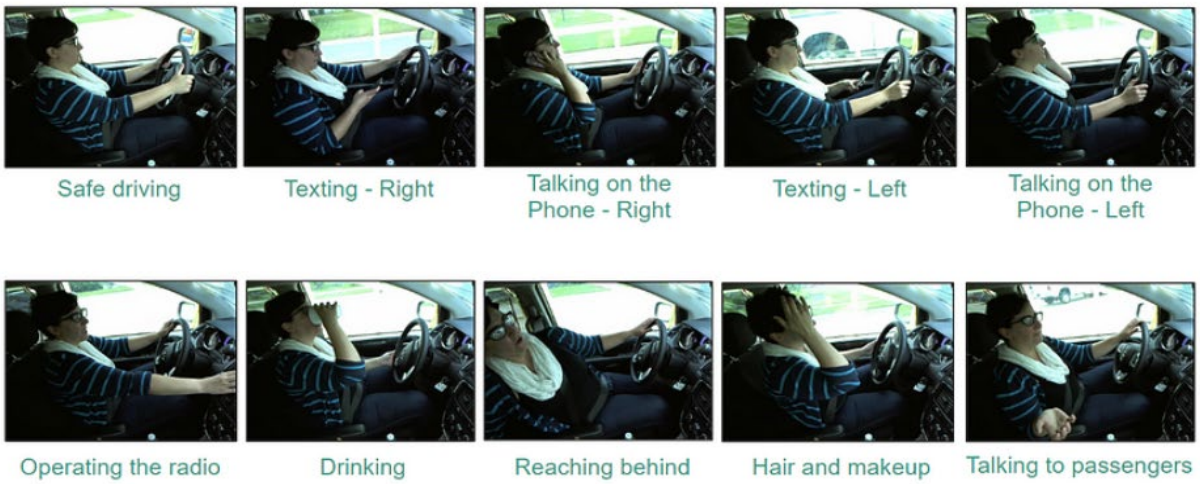
In the paper Machine Learning Techniques for Distracted Driver Detection by Demeng Feng and Yumeng Yue [1], images are resized for training and testing purposes and traditional Machine Learning algorithms are applied namely Linear SVM, SoftMax, Naive Bayes, Decision Tree, and Two-Layer Neural Network. The paper further compared the performance of traditional machine learning algorithms and advanced CNN-based techniques such as ResNet and VGG.

In the paper published by Prof. Kyumin Lee[2], the images were resized and feature extraction techniques were implemented namely Pixel, HOG, Sobel, and Clustering. To implement Principal Component Analysis (PCA) feature extraction for dimensionality reduction was applied. The classification was performed using classifiers like Support Vector Machine (SVM), Decision Tree, Random Forest, 2-layered Neural Network, and Convolution Neural Network (CNN). Each model is subjected to a comparison examination using several feature extraction methods, and accuracy is determined for each method individually. For CNN a Grid Search was performed for individual parameters to obtain the best model and then the transfer learning algorithm VGG-19 was implemented. The paper further discussed using Deep Learning Architectures and combined feature vectors for image representation to obtain results.

In another paper “Driver Distraction Identification with an Ensemble of Convolutional Neural Networks” published by H. Eraqi, Y. Abouelnaga, M. Saad, and M. Moustafa[3], the author preprocesses the images as skin-segmented images, face, and hands images. These along with raw images are trained using a weighted ensemble of CNN.

3. DATASET

State Farm Distracted Driver Detection, available at https://www.kaggle.com/c/state-farm-distracteddriver-detection/data, served as the source of the dataset. The dataset has 10 classes and 22424 total driver images. The 10 classes are safe driving, radio operation, drinking, reaching behind, hair and makeup, talking to passenger(s), texting with the right hand, talking on the phone with the right hand, texting with the left hand, and talking on the phone with the left hand. Each photograph, which is from one of the aforementioned classes, was taken while the driver was engaged in a vehicle-related activity. Each of the colored, 640*480-pixel images is displayed below. The images are reduced in size to 64*64 colored images for testing and training purposes. The dataset is divided into an 80:10 Training-Testing ratio using stratified splitting. The training dataset is further split into a 90:10 Training-Validation set.



3.1 DATA ANALYSIS

For training and testing, the images are resized to 64*64 pixels colored images. Following the application of feature extraction techniques like LBP, HOG, color Histograms, KAZE, and SURF, the extracted features are then subjected to normalization.

Techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are used to reduce dimensionality and avoid the "Curse of Dimensionality." Variance components graphs are used to calculate the n_components of PCA. Following this, ML algorithms are used to obtain accuracy while all the features are stacked together to create a complete image representation.

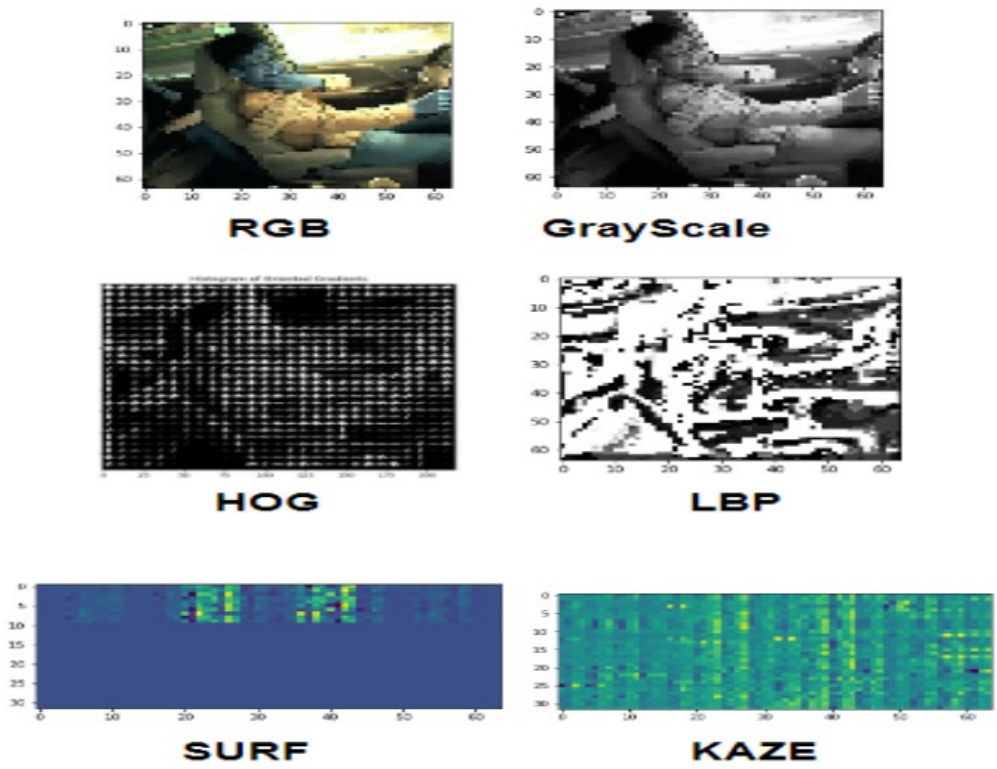


Figure A: Feature Extraction Techniques

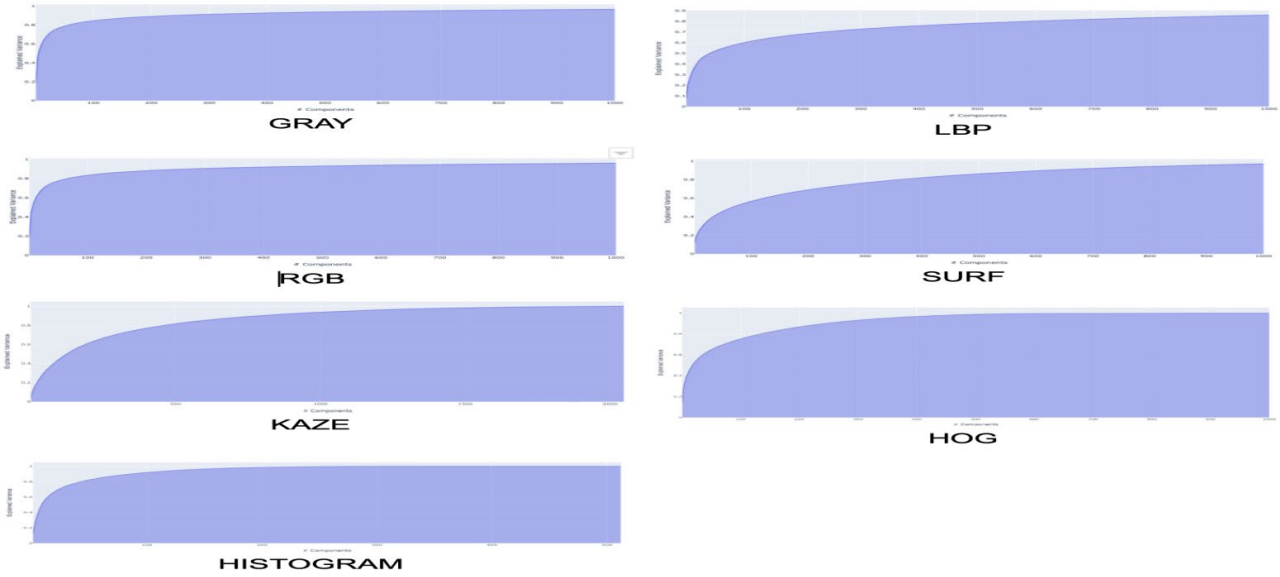


Figure B: PCA variance and n_components graph

4. MACHINE LEARNING METHODS

The following traditional ML algorithms are used for classification and dimensionality reduction.

4.1 CLASSIFICATION METHODS

The following traditional ML algorithms are used to predict the categorical values i.e. identify the category of new observations on the basis of training data.

4.1.1 NAIVE BAYES CLASSIFIER

BAYES' THEOREM

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ≠ 0.

- Basically, we are trying to find the probability of event A, given that event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior probability, i.e. Probability of the event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

With regards to our dataset, we can modify Bayes' theorem by substituting A=y where y is the class variable and B=X where X is the dependent feature vector (of size n).

Assumption of NAIVE BAYES

The basic premise of Naive Bayes is that each feature makes an:

- independent
- equal

contribution to the outcome.

In relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent.
- Each feature is given the same weight (or importance).

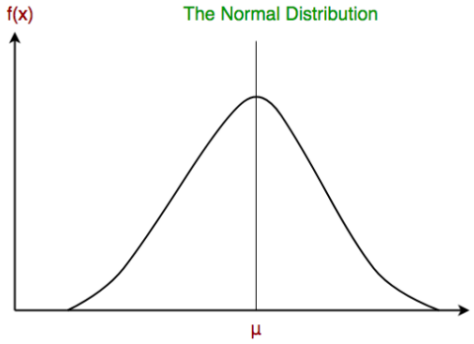
Substituting our naïve assumption to Bayes' theorem i.e. independence among the features, we get the result

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

The method that we discussed above is applicable to discrete data.

Gaussian Naive Bayes Classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell-shaped curve that is symmetric about the mean of the feature values.



The conditional probability in this case is:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Multinomial Naive Bayes

Feature vectors represent the frequencies with which certain events have been generated by a **multinomial distribution**. This is the event model typically used for document classification.

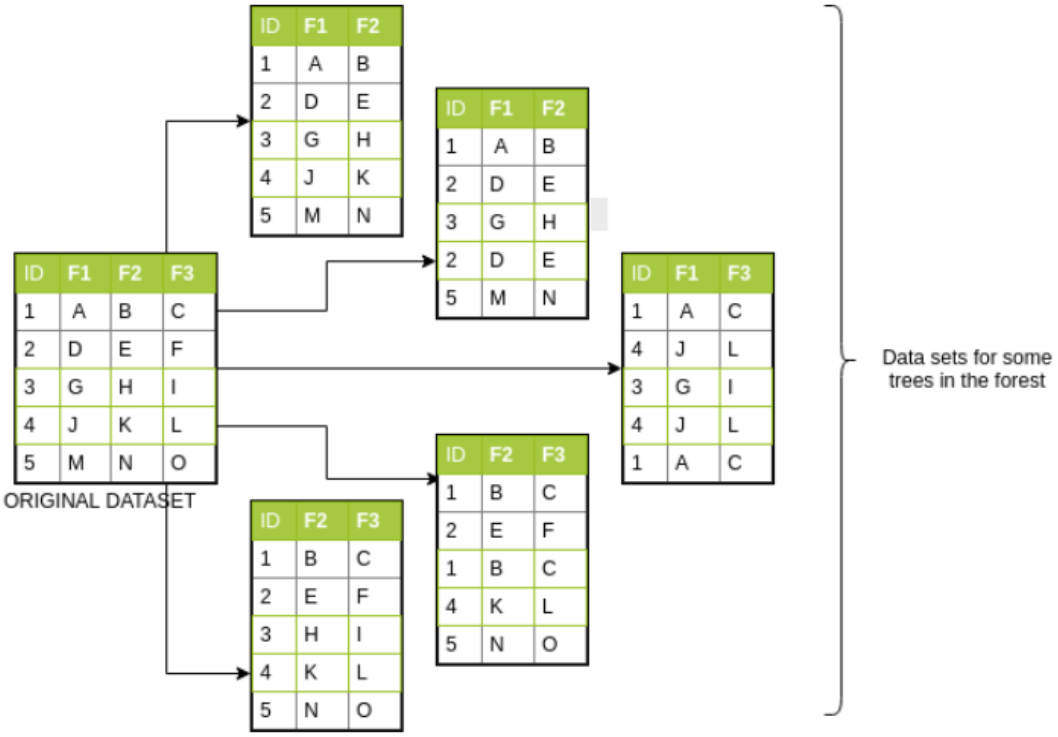
The code to import the Gaussian Naive Bayes classifier in Scikit-Learn is:

```
from sklearn.naive_bayes import GaussianNB
```

4.1.2 RANDOM FOREST CLASSIFIER

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Random forest is a way to reduce overfitting in decision trees and it can also be used to find the importance of features we are using. Each decision tree built will have a randomly selected set of features and a randomly selected set of data points. The number of features in each decision tree in the forest will be less than the total number of features we have in our dataset. So, if we have a feature 'A', it may appear in some of the decision trees of the forest and not in others. Duplication is generally allowed in selecting the data-points from our data-set for the decision trees in the forest.



As shown above, some of the trees have features F1 and F2 but not F3. There is no point in repeating a feature in a decision tree of the forest but as shown data points may be duplicated. This randomness in selecting the features and data-points helps in reducing overfitting. Note that we will make multiple decision trees so that there is very little chance of a feature/data point getting missed out.

BOOTSTRAP AGGREGATION (BAGGING)

Bagging is used when our goal is to reduce the variance of a decision tree. It is a general-purpose procedure for reducing the variance of a predictive model. When applied to trees the basic idea is to grow multiple trees which are then combined to give a single prediction. Combining multiple trees helps in improving precision and accuracy at the expense of interpretation. In bagging, we take multiple smaller data-sets in which we also allow repetition of data points and randomly select some features. Bagging is generally done in reference to data-points.

The code to import the Random Forest Classifier in Scikit-Learn is:

```
from sklearn.ensemble import RandomForestClassifier
```

4.1.3 DECISION TREE CLASSIFIER

A decision tree is a tree-like graph that uses a branching method to illustrate every possible outcome of a decision. The way it is structured, the nodes stand in for where we choose an attribute and where we pose a query. The leaves represent the actual output or class label, while the edges represent the answers to the questions. They work with a straightforward linear decision surface when making non-linear decisions.

Decision trees categorize examples by arranging them in a tree from the root to a leaf node, with the leaf node assigning the example to one of its classifications. The edges descending from each node in the tree represent one of the potential answers to the test case, and each node serves as a test case for a particular attribute. Every subtree rooted at the fresh nodes undergoes this recursive process once more.

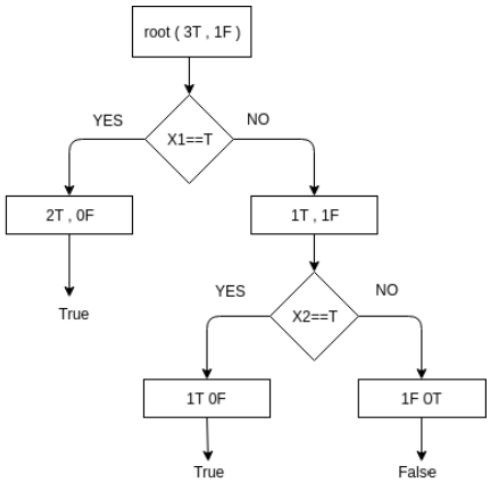
Choosing which feature to use, what condition to use on splitting, and knowing when to stop are all decisions that go into building a decision tree. All features are taken into account in the first split at the root, and the data points are then divided into groups in accordance with this split. Consider that there are n features. At the first level, we will then have n candidate splits. We will now use a function to determine how much accuracy each split will cost us. Data points are split into child nodes based solely on the split (feature) that produces the highest accuracy at this level. The entire tree is created as a result of the recursive division of the child nodes into deeper levels.

We may be able to create an exponential number of decision trees if we have n features. It falls under the NP-HARD Problem category. All tree combinations should be considered in order to determine the best tree. We are therefore interested in learning which tree is good and which is not. We will use the GREEDY approach to reduce costs (while also maximizing accuracy) and build a solid decision tree as a result.

We can understand the decision tree classifier with the example of OR of two variables X1 and X2. The decision tree for the same is shown below:

X1	X2	Y (X1 OR X2)
True	True	True
False	True	True
True	False	True
False	False	False

Truth Table for OR



There are two cases when we need to stop the breaking of nodes into subparts, those are:

1. When we have a pure node, there is no need to proceed further.

2. When all the features are used and we don't have any other feature present.

Advantages of Decision Trees

- Decision trees are easy to comprehend, interpret, and represent.
- Numeral and categorical data can both be handled by decision trees. They can also manage issues with multiple outputs.
- Users need to put in comparatively little work to prepare their data for decision trees.
- The performance of the tree is unaffected by nonlinear relationships between parameters.
- Which fields are crucial for classification or prediction can be clearly seen in decision trees.

Disadvantages of Decision Trees

- Overly complex trees that poorly generalize the data can be produced by decision tree learners. This is called overfitting.
- For estimation tasks where the objective is to forecast the value of a continuous attribute, decision trees are less suitable.
- Decision trees are prone to errors in classification problems with many classes and a limited number of training examples.
- The training of decision trees can be computationally expensive. A decision tree's growth requires extensive computational work. Each candidate splitting field at each node must first be sorted in order to determine which split is best. Some algorithms employ combinations of fields, so it is necessary to look for the best-combining weights. Due to the need to create and evaluate numerous candidate sub-trees, pruning algorithms can also be costly.
- Greedy algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees, where the features and samples are randomly sampled with replacement.
- If some classes predominate, decision tree learners will produce biased trees. As a result, it is advised to balance the data set before fitting it to the decision tree.

4.1.4 K-NEAREST NEIGHBOURS

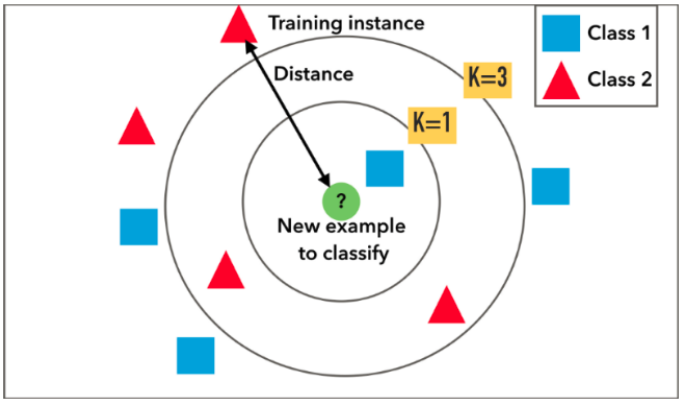
KNN stands for K-Nearest Neighbours. KNN is a straightforward classification algorithm that is frequently used for datasets where data points are divided into several classes and we must predict the class for a new sample point. KNN is a non-parametric and lazy learning algorithm.

Non-parametric means that there are no presumptions about the data distribution made by the algorithm. Non-parametric techniques include those without a fixed model structure and without relying on data from a specific distribution. KNN is thus used as a classification algorithm when there is little information available regarding the distribution of the data.

KNN is referred to as a Lazy algorithm since it does not use training points to do any generalization, which means that there is no separate training phase. KNN retains all of the training data and makes extensive use of it during the testing stage. As a result, KNN does not learn any model; instead, it makes predictions instantly by comparing each training and testing data point.

How to choose the correct value of 'K'?

'K' in KNN is a parameter that refers to the number of nearest neighbours to include in the majority of the voting process.



The test sample (green circle) should either be classified into Class 1(blue squares) or Class 2 (red triangles). The class for the sample testing point is decided based on a majority vote-out. If the value of K is 1, the nearest training point belongs to Class 1, so we will say that the testing sample belongs to Class 1. Now, take the value of K to be 3, again using the methodology of majority vote, we will say that the testing sample belongs to Class 2 (red triangles), since out of the three nearest training data points, two belong to Class 2 and one belong to Class1.

- When K is set to a very low value, such as K = 1 or K = 2, the model may become noisy and exhibit the effects of outliers.
- Inversely, as we increase the value of K, our predictions become more stable due to majority voting/averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing

number of errors. It is at this point we know we have pushed the value of K too far.

- In cases where we are taking a majority vote (e.g picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

KNN has both classification and regression applications.

We will classify the sample point using the technique of majority vote among its neighbours while using KNN for classification - output is a class membership - and the most common class among its K nearest neighbours is assigned to the testing point. Regression's output is an object's property value, which is the average or median of the values of the object's K closest neighbours.

KNN is used to predict a class, which is a discrete value, in classification, whereas KNN predicts continuous values in regression.

We search for points in KNN that are closest to the testing point. If we don't perform feature scaling and one feature is valued in thousands while another is valued in smaller units, the first feature's effect will completely outweigh and take precedence over the second feature's effect in the output. Therefore, it is crucial to apply feature scaling before KNN so that all features contribute equally to the final predicted output for the specified testing point.

Why don't we choose the value of K to be 1?

Choosing the value of K to be 1 makes our model more prone to outliers and overfitting. A value of 1 means that we will consider only the closest (or nearest) neighbor to predict the class for our testing sample, and in the majority of the cases it will lead to overfitting.

Distance Metric for KNN

There are various distance metrics that can be chosen such as Manhattan Distance, Euclidian Distance, etc.

$$Manhattan\ Distance = |\sum_{i=1}^{n} X_1^i - X_2^i|$$

$$Euclidian\ Distance = \sqrt{(\sum_{i=1}^{n}(X_1^i - X_2^i)^2)}$$

$$Minkowski\ Distance = |\sum_{i=1}^{n}(X_1^i - X_2^i)^p|^{\frac{1}{p}}$$

Where $X1$ and $X2$ are two different data points and i traverses over all the features in the given dataset.
The code to import the K-Nearest Neighbours in Scikit-Learn is:

```
from sklearn.neighbors import KNeighborsClassifier
```

Advantages of KNN

- Easy to understand and code.
- Works well for multi-class classification.
- The best K value is insensitive to outliers, but noise and irrelevant features can have an impact on accuracy.
- Versatile, able to be used for both regression and classification.

Disadvantages of KNN

- Computationally expensive, because the algorithm stores all the training data.
- Testing takes a long time.
- Sensitive to scaling and irrelevant features.

KNN becomes biased if the training data split is biased, meaning that the majority of the data points belong to one particular class.

4.1.5 SUPPORT VECTOR MACHINE

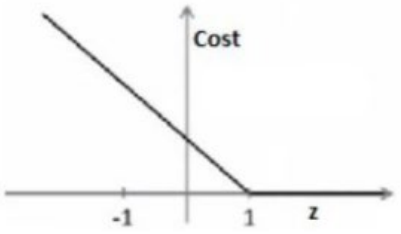
Support Vector Machine (SVM) is a powerful classifier that works both on linearly and nonlinearly separable data. SVM seeks to identify the "best" margin—that is, the separation between the classes—between the line and the support vectors. Unlike logistic regression, which does not, this lowers the chance of data error. The alternative is to use different decision boundaries with weights that are close to the ideal point.

Cost Function

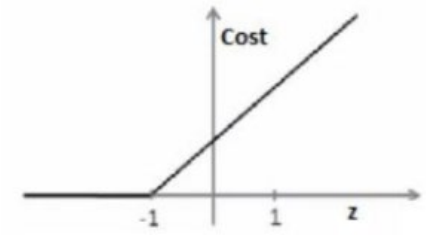
The SVM is trained using the Cost Function. We can make the SVM as accurate as possible by reducing the value of theta.

$$(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

the functions $cost1$ and $cost0$ refer to the cost for an example where y = 1 and the cost for an example where y = 0 respectively. For SVMs, cost is determined by kernel (similarity) functions.



If $y = 1$, we want $\theta^T x \geq 1$ (not just ≥ 0)

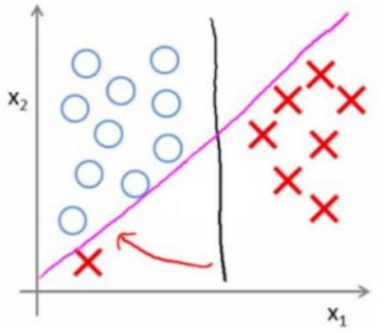


If $y = 0$, we want $\theta^T x \leq -1$ (not just < 0)

Decision Boundary

The boundary between the two classes in an n-dimensional feature space is a plane with n-1 dimensions; in a 2-dimensional feature space, it would be a line, in a 3-dimensional feature space, a plane, and so on. Decision Boundary is also known as a hyperplane.

The objective is to select a decision boundary that has the greatest possible margin between it and any point in the training set, increasing the likelihood that new data will be correctly classified. Some important points to consider before choosing the decision boundary are:



1. SVM would be very sensitive to outliers when only using the large margin.
2. If C is very large, choose the magenta boundary using this quite naive "maximize the margin" strategy.
3. If C is manageably small, continue to use the black decision boundary.

Cross-validation (CV) is a performance statistic used in grid search to improve the C, degree, and other SVM parameters. Finding efficient hyper-parameter combinations is crucial if the classifier is to correctly predict unknown data.

We first divided the available data into k subsets (in most experiments, we set k to be 10), then used a k-fold CV to select C. The remaining k-1 training subsets are used to evaluate one subset after which one subset is used as testing data. Then we calculate the CV error using this split error for the SVM classifier using different values of C, and other parameters. The hyper-parameter combination with the best cross-validation accuracy (or lowest CV error) is chosen and used to train an SVM on the entire dataset.

The code to import the Support Vector Machine in Scikit-Learn is:

```
from sklearn.svm import SVC
```

Advantages of SVM Classifier

- SVM classifiers perform well in high-dimensional spaces and have excellent accuracy.
- SVM classifiers primarily use a subset of training points, which uses very little memory as a result.

Disadvantages of SVM Classifier

- They are ineffective for large datasets in practice due to their lengthy training times.
- Overlapping classes do not perform well with SVM classifiers.

4.1.6 ADABOOST

The machine learning method known as AdaBoost, or adaptive boosting, is applied to classification and regression problems. The core idea behind AdaBoost is to iteratively train a weak classifier on a training dataset while increasing the weight of the misclassified input points in each succeeding classifier. The final AdaBoost model is obtained by combining all the weak classifiers used for training with the weights given to the models according to their accuracy. The strongest model with the highest accuracy receives the highest weight, while the weakest model receives the lowest weight.

The code to import the AdaBoost in Scikit-Learn is:

```
from sklearn.ensemble import AdaBoostClassifier
```

4.1.7 XGBOOST

XGBoost is a distributed gradient boosting library that has been optimized for quick and scalable machine learning model training. A number of weak models' predictions are combined using this ensemble learning technique to create a stronger prediction. Extreme Gradient Boosting, or XGBoost, is one of the most well-known and widely used machine learning algorithms because it can handle large datasets and perform at the cutting edge in many machine learning tasks like classification and regression.

Its effective handling of missing values, which enables it to handle real-world data with missing values without requiring a lot of pre-processing, is one of the key characteristics of XGBoost. Additionally, XGBoost has integrated parallel processing support, allowing user to train models on sizable datasets quickly.

Decision trees are generated sequentially in this algorithm. Weights are significant in XGBoost. Each independent variable is given a weight before being fed into the decision tree that forecasts outcomes. Variables that the tree incorrectly predicted are given more weight before being fed into the second decision tree. These distinct classifiers/predictors are then combined to produce a robust and accurate model. It can be used to solve problems involving regression, classification, ranking, and custom prediction.

The code to import the XGBoost in Scikit-Learn is:

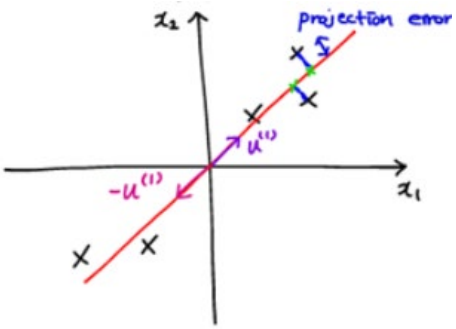
```
import xgboost as xgb
```

4.1.8 OPTIMAL HYPERPARAMETERS VALUES

Model	Optimal Hyperparameters
DT	criterion = 'entropy', max-depth = 20
SVM	C=10 and kernel='rbf'
KNN	n-neighbours = 5
XGB	max-depth = 6, eta = 0.5
Bagging	n-estimators=40
Adaboost	n-estimators=200

4.2 PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a non-parametric, unsupervised statistical method that is largely employed in machine learning to reduce dimensionality. In this method, the objective is specifically to produce a new set of variables that is a linear combination of the initial variables. These new variables are referred to as principal



components.
In order to reduce the squared projection error, PCA seeks out a lower-dimension surface of the line onto which it should project the data. The perpendicular distance from the principal vector is the square projection error.

DIMENSIONALITY REDUCTION

There are frequently too many features in many datasets to draw any conclusions. It becomes more difficult to visualize the training set and then work on it as the number of features increases. Sometimes, the majority of these features are redundant because they are correlated. One can envision a reduction of 2D data with linearly dependent features to a single dimension. Algorithms for dimensionality reduction are useful in this situation.

Advantages of Dimensionality Reduction

- The amount of time and storage space needed is decreased.
- The machine-learning model performs better when multi-collinearity is removed.
- When the data is scaled down to very low dimensions, such as 2D or 3D, it is simpler to visualize the data.

Disadvantages of Dimensionality Reduction

- It might result in some data loss.
- It frequently uncovers linear correlations between variables, which can be problematic.
- It fails in cases where mean and covariance are not enough to define datasets.
- We might not be aware of the optimal number of principal components or dimensions. In actuality, some general guidelines are used.

Application of PCA

1. **Data Compression:** Reducing the data from higher dimensions to fewer dimensions, thus saving space.
2. **Data visualization**: It becomes easier to visualize the data when reduced to very low dimensions such as 2D or 3D.
3. **Increase Speed and Performance**: Removal of multi-collinearity improves the performance of the machine-learning model.

MATH BEHIND PCA

PCA requires the direction in which the data changes the maximum (variance is maximum).

For this, we require the

1. Covariance Matrix
2. Eigen Values and Eigen Vectors: We use SVD (Single Value Decomposition)

SVD is performed on the covariance matrix and we get a set of eigenvectors. These vectors will be unit vectors in the original space.

The **first eigenvector** will be the vector that explains the variance to the maximum (in the best manner), the **second eigenvector** is the unit vector which explains the variance in the second-best manner, and so on.

Eigenvalues represent the value by which a particular direction contains the information.

How to select K (n_components)

The value of k is directly proportional to the variance in the data that can be explained, so picking the right value for k is crucial. Our main goal should be to choose a k that is small enough to explain the majority of the variance in the transformed data before converting the data to k dimensions.

We first use PCA with no n_components value. This indicates that after using PCA, we still have the same number of components. However, all of the component variations have now been explained.

The total variance of the data is the sum of these variances.

Consider the scenario where we choose to keep 3 features after PCA. The percentage of variance we maintain is calculated by dividing the sum of the variances of these three features by the total variance.

The code to calculate the number of n_components based on the amount of variance needed:

```
total = sum(pca.explained_variance_)
k = 0
current_variance = 0
while current_variance/total < 0.99:
current_variance += pca.explained_variance_[k]
k = k + 1
k
```

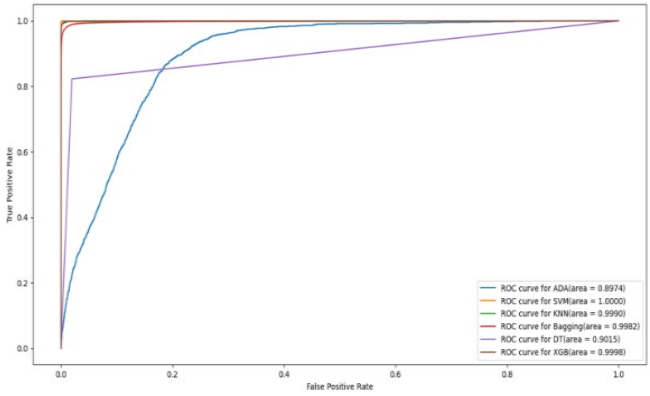
To maintain 99% variance, we need k components i.e. 17 components which is the output of this code.

The code to import the Principal Component Analysis in Scikit-Learn is:

```
from sklearn.decomposition import PCA
```

Model	Precision	Recall	F1 Score	Acc
DT	0.8221	0.8213	0.8214	0.822
SVM	0.9973	0.9973	0.9973	0.997
KNN	0.9872	0.9870	0.9870	0.987
XGB	0.9856	0.9849	0.9852	0.985
Bagging	0.7927	0.7848	0.7861	0.789
Adaboost	0.7197	0.6957	0.7010	0.693

PCA



ROC curve for PCA

*An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds.

4.3 LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis or Normal Discriminant Analysis or Discriminant Function Analysis is a dimensionality reduction technique that is commonly used for supervised classification problems.

LDA makes two assumptions: that the covariance matrices of the various classes are equal and that the data has a Gaussian distribution. Additionally, it assumes that the data can be classified accurately by a linear decision boundary and that the data is linearly separable.

In order for LDA to function, the data are projected onto a lower-dimensional space with a maximum degree of class separation. Finding a group of linear discriminants that

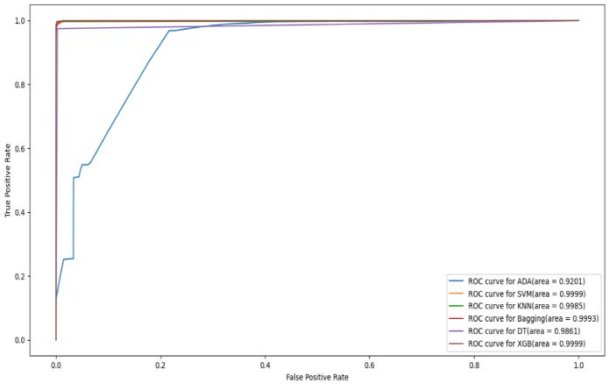
maximize the proportion of between-class variance to within-class variance is how it accomplishes this. In other words, it determines the directions in the feature space that effectively distinguish the various data classes.

The code to import the Linear Discriminant Analysis in Scikit-Learn is:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

Model	Precision	Recall	F1 Score	Acc
DT	0.9753	0.9752	0.9751	0.974
SVM	0.9881	0.9876	0.9876	0.987
KNN	0.9922	0.9924	0.9923	0.992
XGB	0.9912	0.9912	0.9912	0.991
Bagging	0.9825	0.9826	0.9825	0.982
Adaboost	0.5160	0.5785	0.5191	0.574

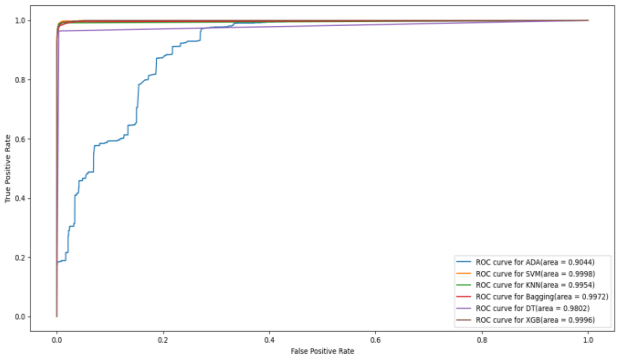
LDA



ROC curve for LDA

Model	Precision	Recall	F1 Score	Acc
DT	0.9638	0.9639	0.9638	0.964
SVM	0.9799	0.9791	0.9795	0.979
KNN	0.9806	0.9793	0.9799	0.979
XGB	0.9757	0.9756	0.9756	0.976
Bagging	0.9720	0.9710	0.9714	0.971
Adaboost	0.6880	0.6634	0.6304	0.658

LDA on PCA



ROC curve for LDA on PCA

4.4 TRANSFER LEARNING

A peculiar phenomenon is shared by many deep neural networks trained on images: early layers of a deep learning model attempt to learn low-level features, such as detecting edges, colours, variations in intensities, etc, such features don't seem to be specific to a particular dataset or task. In both situations, we must find these inconspicuous features. Regardless of the cost function or image dataset, all these features exist. Thus learning these features in one task of detecting lions can be used in other tasks like detecting humans.

Transfer learning is a deep learning technique where a model that has been trained on one task is adapted to fit another task. When the second task is similar to the first task or when there is a dearth of data for the second task, this can be helpful. The model can learn more quickly and efficiently on the second task by starting with the learned features from the first task. Because the model will have already picked up general

features that are likely to be helpful in the second task, this can also aid in preventing overfitting.
Using the State Farm dataset of distracted drivers, there are three general methods for retraining the pre-trained models.

1. Retraining the pre-trained model should only be done on the final classifier layer. In exchange for model accuracy, this is the simplest strategy to implement and requires the least amount of time and computing power.
2. Retrain the classifier layer and the last few model layers. This approach requires more time to execute but could lead to increased accuracy and computational power.

Retrain the model in its entirety from scratch. Because it requires the most time and computational resources, this method is the least preferred one.

4.5 CONVOLUTION NEURAL NETWORKS

Neural networks are a subset of machine learning, and they are at the heart of deep learning algorithms. They're made up of node layers, each of which has an input layer, one or more hidden layers, and an output layer. Each node is connected to the others and has a weight and threshold assigned to it. If a node's output exceeds a certain threshold value, the node is activated, and data is sent to the next layer of the network. Otherwise, no data is passed on to the network's next layer. While the majority of that article concentrated on feedforward networks, there are numerous more neural network types that are employed for a variety of use cases and data kinds. Convolutional neural networks (ConvNets or CNNs) are more typically employed in classification and computer vision applications compared to recurrent neural networks, which are regularly utilized in speech recognition and natural language processing. Before CNNs, feature extraction approaches that required manual labour were used to identify objects in photos. On the other hand, convolutional neural networks make use of linear algebra concepts to offer a more scalable solution to picture classification and object recognition problems.

How Convolutional Neural Network Works

The superior performance of convolutional neural networks with image, speech, or audio signal inputs sets them apart from other neural networks. They are divided into three types of layers:
1) Convolutional layer
2) Pooling Layer
3) Fully connected layer

The convolutional layer is the top layer of a convolutional network. The fully-connected layer is the last layer, though extra convolutional layers or pooling layers may be added following convolutional layers. With each layer, the CNN gets more complex, detecting larger portions of the image. Early layers emphasize fundamental elements like colours

and borders. The complexity of the image data increases as it moves through the CNN layers.

Convolutional Layer

Given that it is where the majority of the processing occurs, the convolutional layer is the most crucial part of a CNN. Along with other things, it needs input data, a filter, and a feature map. Assume for the moment that the input is a colour image that is composed of a 3D matrix of pixels. Height, width, and depth, which stand for the RGB colour space in an image, will therefore be the three dimensions of the input. The receptive fields of the image will be traversed by a feature detector, also known as a kernel or filter, which will look for the feature's presence. This method is referred to as convolution.

The feature detector is a two-dimensional (2-D) weighted array that represents a portion of the image. The filter size, which can vary in size, is usually a 3x3 matrix, which also determines the size of the receptive field. After that, the filter is applied to a portion of the image, and a dot product between the input pixels and the filter is calculated. After that, the dot product is fed into an output array. The filter then shifts by a stride, and the process is repeated until the kernel has swept across the entire image. A feature map, activation map, or convolved feature is the final output of a series of dot products from the input and the filter.

Pooling Layer

A dimensionality reduction approach called downsampling, often referred to as pooling layers, lowers the number of parameters in the input. Similar to a convolutional layer, the pooling process sweeps a filter across the entire input, but this filter lacks weights. Instead, the kernel populates the output array from the values in the receptive field using an aggregation function. Pooling can be divided into two categories:

Max pooling: As the filter passes across the input, it chooses the pixel with the highest value to deliver to the output array. This method is applied more frequently than average pooling.

Average pooling: As it passes across the input, the filter computes the average value within the receptive field and transmits it to the output array.

While the pooling layer loses a lot of information, it does have a few advantages for CNN. They assist in reducing complexity, increasing efficiency, and reducing the risk of overfitting.

Dropout Layer

Adding dropout layers is a regularization method to prevent overfitting. A dropout layer eliminates some feature detectors by randomly dropping out some activations and setting some unit activations to zero. This reduces some of the high variance that results from it.

Fully Connected Layer

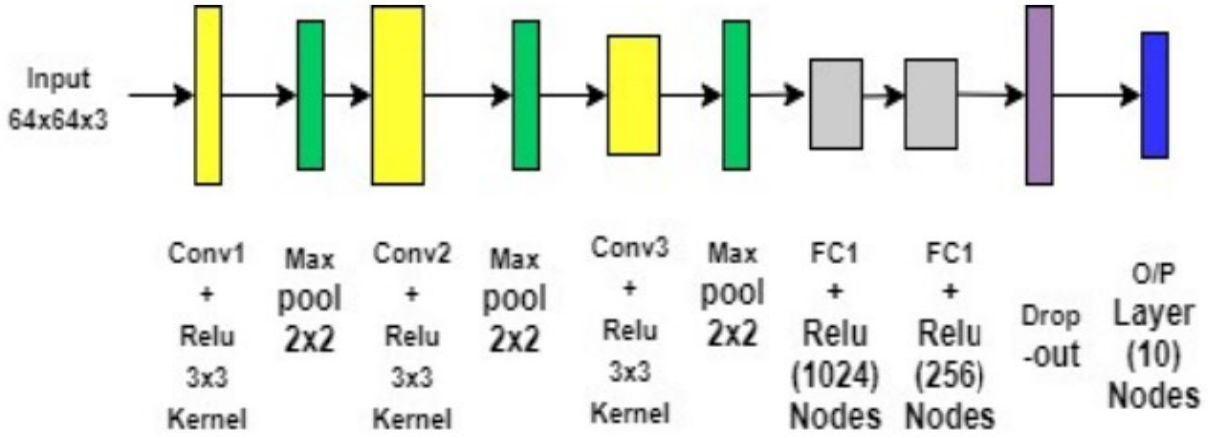
The name of the layer, full-connected, is self-explanatory. As mentioned earlier, with partially linked layers, the input image's pixel values are not directly connected to the output layer. On the other hand, every node in the output layer has a direct connection to a node in the layer below it in the fully connected layer.

Based on the features retrieved by the previous layers and their respective filters, this layer performs classification tasks. In contrast to FC layers, which commonly utilize a softmax activation function to generate a probability ranging from 0 to 1, convolutional and pooling layers frequently use ReLU functions to categorize inputs.

Types of Convolutional Neural Network

1. AlexNet
2. VGGNet
3. GoogleNet
4. ResNet
5. ZFNet

CNN ARCHITECTURE



This architecture of Convolution Neural Networks is used in the classification process of input images. The CNN architecture involves multiple layers of operations performed on the input images of size (64,64,3).

Rectified Linear Units (ReLU) Layers

To increase the model's nonlinearity, the ReLU layer applies an element-wise nonlinear activation function. The ReLU function is defined as $F(x) = \max(0, x)$.

The ReLU function's primary advantage over other activation functions such as step, and sigmoid functions is that it does not simultaneously activate all of the neurons i.e. If the input is negative, as shown by the ReLU function, it will be converted to zero and the neuron will not be activated.

Residual Network (ResNet-101)

A Vanishing/Exploding gradient is a phenomenon that occurs when the CNN

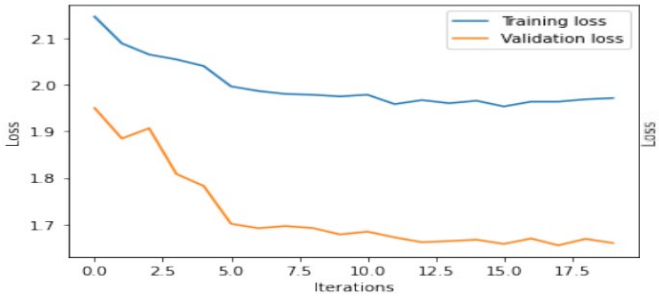
architecture uses more layers in a deep neural network to minimize the error rate, which makes the gradient become 0 or excessively large and raises the training and testing error rates.

To counter this problem, we employ a method known as skip connections. The skip connection bypasses some layers in between to link layer activations to subsequent layers. This creates a leftover block. These residual blocks are stacked to create ResNets. The strategy behind this network is to let the network fit the residual mapping rather than have layers learn the underlying mapping.

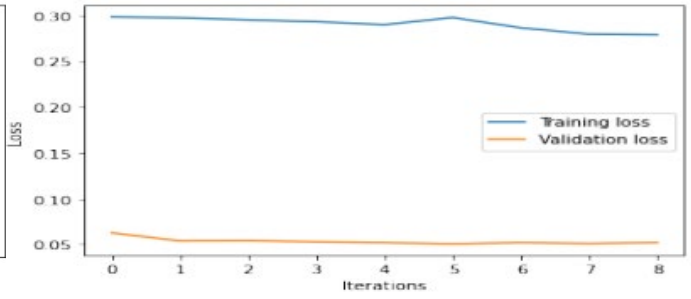
The benefit of including this kind of skip connection is that regularization will skip any layer that degrades architecture performance. As a result, training a very deep neural network is possible without encountering issues with vanishing or exploding gradients.

Two strategies used with ResNet-101 are:

1. **Strategy S1:** Only the final classifier layer of the pre-trained model needs to be retrained, and all other parameters are frozen.
2. **Strategy S2:** Retrain the last few layers of the model including the classifier layer.



Training and Validation loss for S1



Training and Validation loss for S2

Model	Precision	Recall	F1 Score	Acc
CNN	0.98	0.98	0.98	0.98
ResNet-101(S1)	0.50	0.46	0.42	0.47
ResNet-101(S2)	0.99	0.99	0.99	0.99

Deep Learning Model

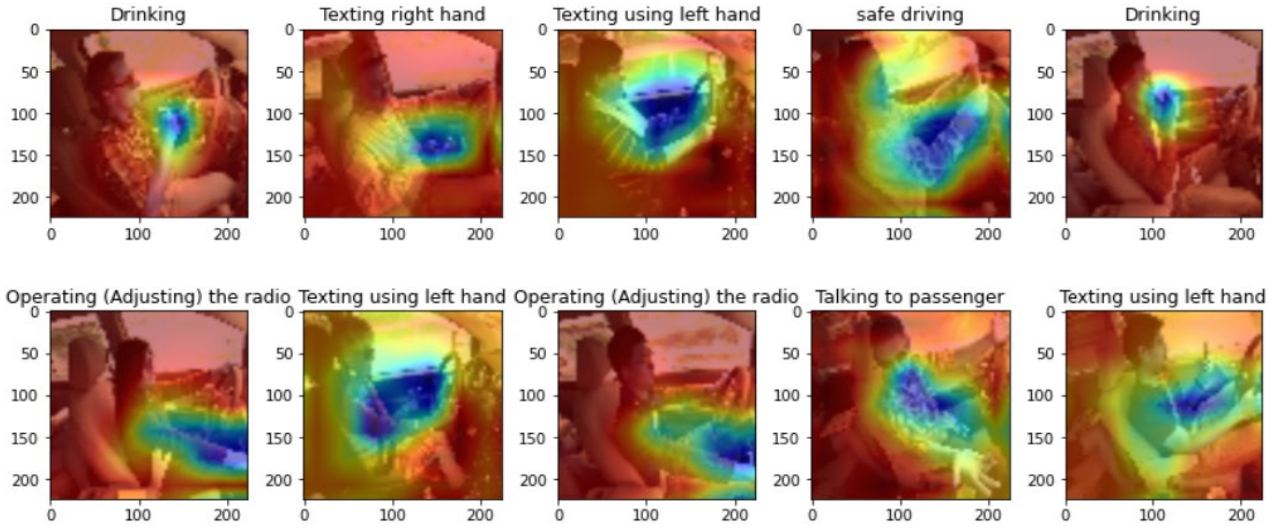
Comparing the two strategies we find that strategy 2 performs better with ResNet-101

ResNet Prediction Visualisation

4.6 CLASS ACTIVATION MAPPING

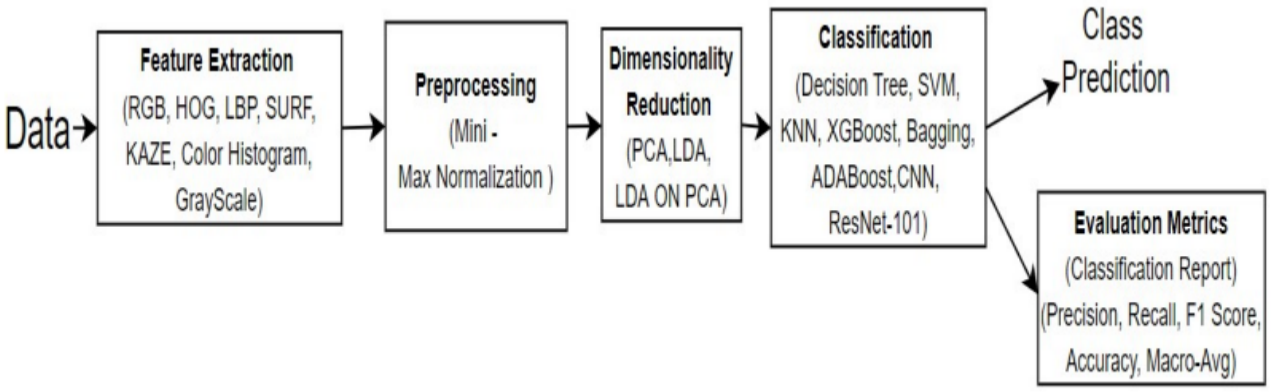
Class activation maps (CAMs) are a way to draw attention to the areas in an image that a CNN uses to determine the image's classification. The result of a CAM is a heat map that is visually superimposed over the image that the neural network is analyzing. The parts of the image that the neural network is concentrating on are represented by the areas of greater intensity. ResNet (Residual Network) is a popular type of CNN architecture, and CAM can be applied to ResNet models to generate class activation maps.

To generate Class Activation Maps (CAM) for a ResNet model, start by loading a pre-trained ResNet model and removing the final classification layer. Access the network's intermediate feature maps next, frequently selecting the final convolutional layer prior to global average pooling. Apply class-specific weights to the feature maps using the final classification layer, then perform global average pooling. Create a class activation map that highlights the regions pertinent to the selected class by upsampling the weighted feature vector to the original image size. Finally, user can see which areas most significantly influenced the model's classification choice by superimposing the CAM on the original image. Depending on the deep learning framework being used, such as PyTorch or TensorFlow, the specific implementation may change.



Class Activation Map for ResNet-101(S2)

5. ARCHITECTURE IMPLEMENTED



6. INFERENCES AND OBSERVATIONS

- PCA outperformed LDA and LDA over PCA
- Better accuracy is obtained by combining features from different feature extraction methods than by using individual features.
- As boosting techniques work better on datasets with high variance than our data set, which has low variance, AdaBoost performed poorly for classification on this dataset.
- SVM had the highest AUC-ROC score after the ROC curves for various conventional models were plotted.
- ResNet-101 S2 outperformed ResNet-101 S1, which also had higher accuracy than that attained using CNN.
- Techniques for data extraction and augmentation improve the performance of the classifier.

7. CONCLUSION

The development and implementation of distracted driver detection systems represent a significant milestone in the ongoing efforts to improve road safety. Artificial intelligence-based technology like this one has the potential to lessen the serious effects of distracted driving. As this report has made clear, distracted driving is a major factor in collisions and fatalities on our roads. The effects of distracted driving extend throughout our society, having an impact not only on the victims and their families but also on the general economy and healthcare system.

The distracted driver detection system, discussed in this report, has shown promising results in identifying and alerting drivers when they engage in activities that divert their attention from the road. These systems have the potential to be extremely useful in averting accidents, saving lives, and lessening the financial burden brought on by traffic accidents. Additionally, they encourage safer and more responsible driving habits, which are crucial in our quickly paced and globally connected world.

The significance of these systems cannot be overstated as we look to the future. It is more important than ever to have solid and trustworthy mechanisms in place to monitor driver behavior due to the development of autonomous vehicles and the ongoing integration of technology into our daily lives. Distracted driver detection systems are helpful for today's cars, but they will also be essential for the transition to autonomous driving, where human drivers may still be required to take over in some circumstances.

8. REFERENCES

DATASET: https://www.kaggle.com/competitions/state-farm-distracted-driver-detection/data

Research Papers [1]:
Demeng Feng, Yumeng Yue, “Machine Learning Techniques for Distracted Driver Detection”

Research Paper [2]:
Kyumin Lee, “Machine Learning: Distracted Driver Detection”

Research Paper [3]:
H. Eraqi, Y. Abouelnaga, M. Saad, and M. Moustafa, “Driver Distraction Identification with an Ensemble of Convolutional Neural Networks”

Deepthi M. Pisharody “Driver distraction detection using machine learning techniques”

Satya Naren Pachigolla “Distracted Driver Detection using Deep Learning”

Fabio Tango and Marco Botta “Real-Time Detection System of Driver Distraction Using Machine Learning”