Sandia Project

Name: Ronald Mosness

MACHINE LEARNING & SIGNAL ANALYSIS TO AUTOMATICALLY PREDICT COMPONENT DEGRADATION

INTRODUCTION

Signals are a means of communication or data transfer. They are present in every kind of system in the world where interactions are required. These systems can be the equipment and devices we use, our biological systems, the geological system, and so on.

Signals can be either discrete (signals that exist only at distinct points in time or space) or continuous (signals that exist across an uninterrupted range of time or space). Commonly, digital signals are discrete while analog signals are continuous. Digital signals are used extensively in modern electronics, communication systems, and computing devices. In electronics, analog signals are used to represent real-world phenomena such as sound, light, or temperature.

Noise is a common phenomenon that disturbs any signal. Any type of disruption to the signal can be termed as noise. This can be faulty hardware, background signals or radiation, electrical signals due to improper grounding, biological signals, audio, as well as useful signals from other equipment. Analog signals are more susceptible to noise than digital signals. This is why usually hardware components use analog-to-digital converters (ADCs). While this conversion eases the process of signal computation, they have some drawbacks. ADCs discretize analog signals into a finite number of digital levels, introducing quantization error. This error occurs because the ADC can't perfectly represent the infinite range of analog voltages with a finite number of digital values. Higher-resolution ADCs can mitigate this error, but they tend to be more complex and expensive. Moreover, ADCs are sensitive to noise, both from the input signal and internal sources such as thermal noise and quantization noise. Noise can degrade the accuracy and reliability of the digitized signal, particularly in low-amplitude or high-precision applications.

When hardware components degrade over time, they produce signals that are substandard or degraded. These signals go unnoticed until the device completely malfunctions while producing data that might be compromised. An example can be a faulty EEG acquisition headset. EEG or electroencephalography is a neuroimaging technique that captures the electrical signals generated from the brain. An EEG headset makes use of electrodes (special kinds of sensors) to acquire these signals. EEG signals are quite small in amplitude (in microvolts) and so require higher sensitivity and responsivity of the device to detect them. They are also highly dynamic in nature. A faulty EEG headset may incorporate extra noise to the already noisy signal (due to impedance, muscle movements, and heart signals). While the other noises are taken into account by health professionals and neuroscientists, the device noise may get overlooked and can pose a danger to the diagnosis.

Physically testing the hardware components can be an extensive process and requires time, energy, and finance. Literature suggests that AI can be a wonderful tool for signal analyses and has proved to be helpful in the classification of different signals. Many machine learning and deep learning models have been made for the classification of EEG signals as well, usually to classify EEG signals of healthy individuals from neurodeficit individuals (having abnormal neurological function). The general aim of this project is to find an innovative way components could be tested automatically by checking the signal outputs they produce and comparing them to a dataset of “healthy” signals by using Machine

Learning. Specifically, the objective of this report is to classify degraded EEG signals from healthy ones using four different machine-learning models.

METHODOLOGY

Generating signals

The signals used in this study are EEG signals which were generated by using Python 3.8. Two types of signals were generated, healthy EEG signals and degraded EEG signals. At first, eight healthy and eight degraded EEG signals were generated via the NumPy toolbox using Python. Since EEG waves are emitted from the brain, they are sinusoidal in nature and superimposed with different noises which can be due to the signal disturbances from the rest of the biological systems (such as the heart), the impedance of the signal acquisition system, and background signals. For this purpose, random noise was added to the signals to make them as realistic as possible. For this study, only a single-channel EEG signal was sufficient, with a sampling frequency of 1000 Hz. Each signal was 10 seconds long.

For generating the degraded signals, all parameters were kept the same. Additionally, to 'degrade' the signal, Gaussian noise was added. For 4 signals, the amplitude for Gaussian noise was kept at 0.5 whereas for the remaining 4, it was kept at 0.9.

Later on, during the first trial of taking the signals as input in the Machine Learning models, overfitting was noticed so the number of signals was increased to eighteen for each type.

Code for generating signals:

```
import numpy as np
import matplotlib.pyplot as plt
def generate_dummy_eeg(duration=10, sampling_rate=1000, noise_level=0.5):
    time_points = np.arange(0, duration, 1/sampling_rate)
    # Generate random noise
    noise = np.random.normal(0, noise_level, len(time_points))
    # Create a sine wave resembling brainwave patterns
    theta = 8  # Frequency in Hz
    sine_wave = np.sin(2 * np.pi * theta * time_points)
    # Combine noise and sine wave to simulate EEG signal
    eeg_signal = noise + sine_wave
    return time_points, eeg_signal
duration = 10  # seconds
sampling_rate = 1000  # Hz
noise_level = 0.5time_points1, eeg_signal1 = generate_dummy_eeg) #generation of a single signal
```

Code for the addition of Gaussian noise to generate degraded signals:

```
noise_amplitude = 0.9  # Adjust the noise amplitude as needed

gaussian_noise = np.random.normal(0, noise_amplitude, len(eeg_signal1))

eeg_with_noise1 = eeg_signal1 + gaussian_noise    #add noise to the generated signal
```

It is important to note that the same healthy signals were not degraded, but new signals were created for the purpose of degrading them to remove any biased readings.

Feature extraction

EEG features become more robust in the frequency domain. More importantly, when dynamic signals are decomposed into power spectrums, they become more easily understandable, especially for machine learning purposes. For this reason, the Welch periodogram was used to compute the power spectral densities of the data. The SciPy toolbox of Python was incorporated. The parameters of the welch function were:

A 4-sec window length

50% overlapping

A 'Hanning' window type

These parameters were set for both, healthy and degraded signals. As a result, each 10-second signal produced 2001 power spectral densities.

Code for applying Welch periodogram:

```
import scipy

from scipy import signal

import numpy as np

import pandas as pd

healthy_data1 = pd.read_csv(r'C:/Users/PC/Desktop/sandra/data_healthy/eegh9.csv', header=None)

healthy_data1 = np.transpose(healthy_data1)

raw_data1 = np.array((healthy_data1))

fs = 1000

wl = fs*4

pwelch1 = signal.welch(raw_data1,fs=1000,window='hann',nperseg=wl,noverlap=wl*0.5,nfft=wl)

hsig1 = pwelch1[1]

np.savetxt('healthy_w9.csv', hsig1, delimiter=',')
```

Data analysis

To check whether or not the data of both types was drastically different, the mean power spectral densities of 8 signals for healthy and degraded were calculated respectively. Since the signals were generated using a normal distribution, parametric tests could be performed on them. To compare the healthy signals with degraded ones, t-test was applied. This t-test was two-sample, assuming unequal variances, and two-tailed. The results show that there is no significant difference between the power spectral densities of the healthy signals and the degraded signals ($p > 0.05$) so it can be said that it is very difficult to differentiate between the two types of signals using normal analyses, so the idea of using machine learning is a necessary intervention.

	healthy	*degraded*
Mean	0.001486	0.002507
Variance	0.000983	0.00095
Observations	2001	2001
Hypothesized Mean Difference	0	
df	3999	
t Stat	-1.03899	
P(T<=t) two-tail	0.298871	

Table 1: t-Test between the mean healthy and mean degraded signals: Two-Sample Assuming Unequal Variances

Data Arrangement

Data was merged into a single array of 36 x 2002, where the rows denoted the number of signals and columns (1-2001) denoted the Power Spectral Densities. A single column of 'labels' was added to the data. This column had two labels, 'healthy' and 'degraded'. The merged data was shuffled and divided into 80% training and 20% testing data

Variables - TrainingData

TrainingData

26x2002 table

	1989 ar1989	1990 Var1990	1991 Var1991	1992 Var1992	1993 Var1993	1994 Var1994	1995 Var1995	1996 Var1996	1997 Var1997	1998 Var1998	1999 Var1999	2000 Var2000	2001 Var2001	2002 category
1	.4664e-04	4.5561e-04	5.9619e-04	2.1195e-04	0.0011	0.0012	5.1108e-04	9.9037e-04	0.0015	0.0013	0.0012	6.0441e-04	1.6885e-04	Healthy
2	.3711e-04	3.1290e-04	5.0682e-04	3.1978e-04	7.3484e-04	7.3984e-04	5.8662e-04	2.4330e-04	4.9926e-04	0.0010	9.4487e-04	0.0011	7.9291e-04	Healthy
3	.3286e-04	5.2825e-04	2.5514e-04	1.7283e-04	1.9754e-04	1.3452e-04	3.1911e-04	4.0556e-04	2.5069e-04	2.5205e-04	3.5046e-04	3.1984e-04	1.8605e-04	Healthy
4	.5838e-04	1.3277e-04	3.2601e-04	9.2534e-04	5.3649e-04	1.9233e-04	6.6557e-04	8.7278e-04	7.1873e-04	5.2637e-04	2.7944e-04	2.4913e-04	1.6310e-04	Healthy
5	.1688e-04	4.9589e-04	7.8071e-04	0.0011	7.9896e-04	4.0682e-04	4.7944e-04	1.8562e-04	2.8486e-04	2.5891e-04	5.2999e-04	5.0766e-04	1.1355e-04	Healthy
6	.7996e-04	3.7775e-04	0.0011	7.2200e-04	0.0011	9.7402e-04	6.5825e-04	8.8281e-04	7.3351e-04	4.6160e-04	8.3145e-05	1.2217e-04	9.4763e-05	Healthy
7	.8664e-04	1.9148e-04	1.4006e-04	1.9760e-04	5.7487e-04	2.4384e-04	3.0578e-04	4.6871e-04	2.8808e-04	3.5402e-04	3.0885e-04	2.7780e-04	1.4182e-04	Healthy
8	.6839e-04	2.4752e-04	2.3972e-04	2.3049e-04	2.3224e-04	4.1947e-04	3.5049e-04	2.2506e-04	3.8564e-04	4.3634e-04	5.6627e-04	5.4928e-04	4.1340e-04	Healthy
9	0.0019	0.0013	0.0012	0.0020	0.0045	0.0034	0.0010	0.0016	9.8733e-04	8.0607e-04	0.0019	0.0029	0.0012	Degraded
10	0.0023	0.0011	0.0012	0.0024	0.0047	0.0028	0.0012	0.0037	0.0018	7.9228e-04	0.0017	0.0018	9.4943e-04	Degraded
11	0.0022	6.7210e-04	0.0013	0.0022	0.0072	0.0039	0.0011	0.0044	0.0016	0.0019	0.0029	0.0015	7.6620e-04	Degraded
12	0.0015	0.0012	0.0021	0.0045	0.0060	0.0037	0.0023	0.0033	0.0016	0.0014	0.0016	0.0019	0.0011	Degraded
13	0.0012	8.9663e-04	5.8698e-04	8.6379e-04	0.0017	9.9432e-04	0.0010	9.4543e-04	0.0016	0.0017	0.0014	0.0019	5.5036e-04	Degraded
14	0.0014	0.0012	6.8928e-04	6.7457e-04	8.9533e-04	0.0014	0.0013	0.0016	0.0014	0.0012	8.1900e-04	8.1977e-04	2.9338e-04	Degraded
15	.9220e-04	2.3247e-04	9.0549e-04	8.1509e-04	3.0872e-04	3.0780e-04	2.9228e-04	1.8449e-04	3.5070e-04	7.6576e-04	3.6811e-04	1.2468e-04	3.3278e-05	Healthy
16	.3387e-04	3.0906e-04	1.7624e-04	1.8895e-04	6.4515e-04	3.2565e-04	6.3726e-04	5.5481e-04	2.5540e-04	3.3692e-04	4.7867e-04	4.6342e-04	1.5529e-04	Healthy
17	.4107e-04	1.8559e-04	2.9287e-04	4.5037e-04	6.5369e-04	6.9951e-04	6.9761e-04	2.7787e-04	4.4889e-04	5.3064e-04	4.2315e-04	8.2776e-04	3.0170e-04	Healthy
18	.9154e-04	3.2696e-04	3.3288e-04	4.6571e-04	4.4626e-04	3.3601e-04	7.2383e-04	4.3626e-04	1.9055e-04	4.0699e-04	2.6015e-04	1.5597e-04	1.4831e-04	Healthy
19	.2874e-04	2.9461e-04	2.5463e-04	2.5991e-04	3.3535e-04	4.1952e-04	6.2959e-04	5.7101e-04	4.7731e-04	1.6193e-04	2.1830e-04	4.6064e-04	3.2282e-05	Healthy
20	.7850e-04	6.5285e-04	7.5331e-04	0.0012	7.5263e-04	4.3264e-04	5.5254e-04	6.0192e-04	5.4461e-04	1.6078e-04	5.0058e-04	9.0444e-04	5.9836e-04	Healthy
21	0.0030	8.9352e-04	9.8454e-04	0.0016	0.0010	0.0013	0.0026	0.0046	0.0020	0.0026	0.0030	0.0019	7.9179e-04	Degraded

Figure 1: Data arrangement of training data

Building the Model

All classification tools and machine learning algorithms were developed using MATLAB R2021a (The MathWorks Inc., Natick, 2021). Four built-in classifiers were used within the Classifier Learner app on MATLAB, namely, Support Vector Machine (SVM), Ensemble, K Nearest Neighbor (KNN), and Artificial Neural Network (ANN). These classifiers were chosen after surveying various research papers based on the classification of EEG signals.

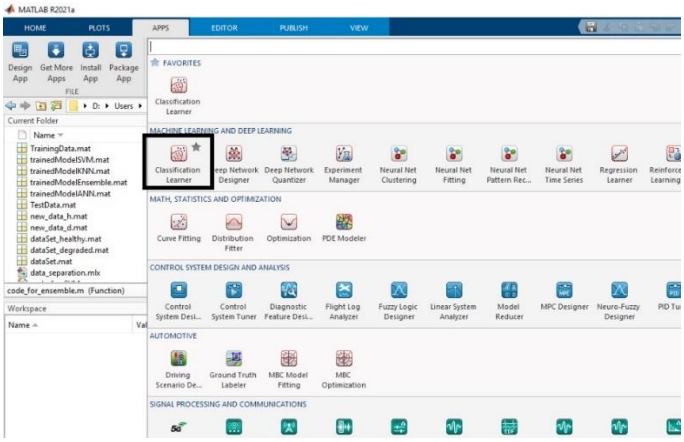

Figure 2: Classification Learner application module in MATLAB

The validation method used was the K-fold method with 5 folds and 2 iterations. The k-fold method (in this case 5-fold) is a cross-validation method which makes five subsets of the dataset and uses four subsets for training and one for validation. This method is repeated five times until all the subsets have been validated. The k-fold validation method is used to remove over-fitting. Overfitting is a common problem in machine learning where a model learns to capture the noise or random fluctuations in the training data rather than the underlying pattern or relationship. One of the challenges that are faced when applying ML to EEG data is the higher chances of overfitting, therefore, cross-validation was performed.

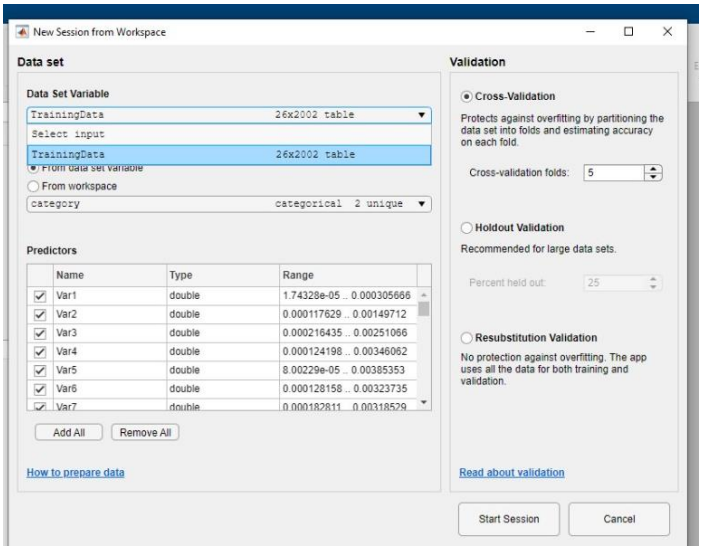

Figure 3: Data entry and validation

Parameters of models

Artificial neural network (ANN)

Artificial neural networks are versatile and powerful tools in machine learning, capable of learning complex patterns and relationships in data and making accurate predictions across a wide range of applications. ANNs are used widely in classifying EEG signals, especially in the frequency domain or in the form of power spectral density. The activation function used was ReLU. An activation function introduces nonlinearity into the network, allowing it to learn and model complex relationships in the data. ReLU, which stands for Rectified Linear Unit, is one of the most commonly used activation functions in ANN due to its simplicity. One fully connected layer was used with five neurons.

Support Vector Machine (SVM)

SVMs are a supervised learning algorithm used for classification and regression tasks, although they're primarily known for their effectiveness in classification. SVMs are popular for EEG signal classification due to their ability to handle high-dimensional data and their robustness against overfitting. Two iterations were used; the kernel function was set to linear. Bayesian optimizer was used and the multiclass method applied was one vs one.

K nearest neighbour (KNN)

KNN is considered a non-parametric, lazy learning algorithm, meaning that it doesn't make any assumptions about the underlying data distribution and doesn't learn a specific model during training. Instead, it memorizes the entire training dataset and makes predictions based on the similarity between new data points and the existing training examples. KNN is a simple and intuitive algorithm that is used in EEG classification for its ease of implementation and effectiveness, especially in scenarios where the underlying data distribution is not well understood. The hyper-parameter values were set to four neighbours, Minkowski (cubic) distance metric, and inverse distance weight.

Ensemble

An ensemble is a technique that combines the predictions of multiple individual models (often called base learners or weak learners) to produce a more accurate and robust prediction. The idea behind ensemble learning is that by combining the predictions of multiple models, the weaknesses of individual models can be mitigated, and overall performance can be improved. Ensemble learning techniques can be effectively applied to EEG (electroencephalography) classification tasks to improve prediction accuracy and robustness. The ensemble method used here was LogitBoost. This method specifically targets improving the performance of binary classification tasks by directly modeling the class probabilities.

RESULTS AND CONCLUSION

Table 2: Validation and testing accuracies

Accuracy	ANN	SVM	KNN	Ensemble
Validation	96.2%	100%	80.8%	88.5%
Testing	100%	100%	80%	90%

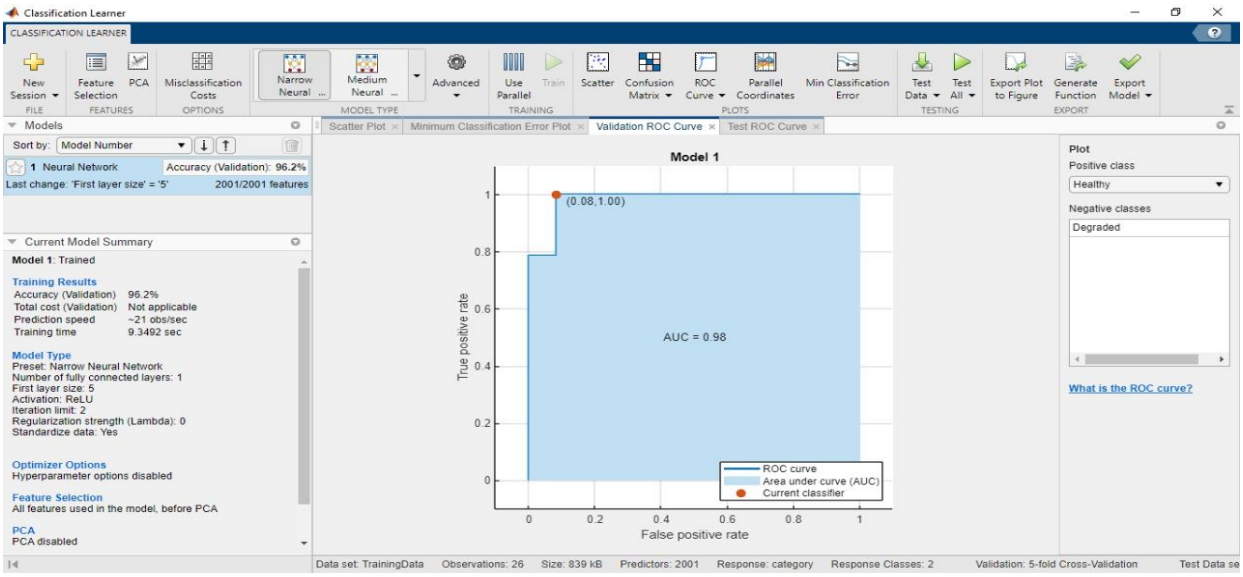

Figure 4: ANN validation details and ROC curve

ROC curve stands for Receiver Operating Characteristic curve. It's a graphical representation commonly used to evaluate the performance of binary classification algorithms. The ROC curve visualizes the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) across different decision thresholds.

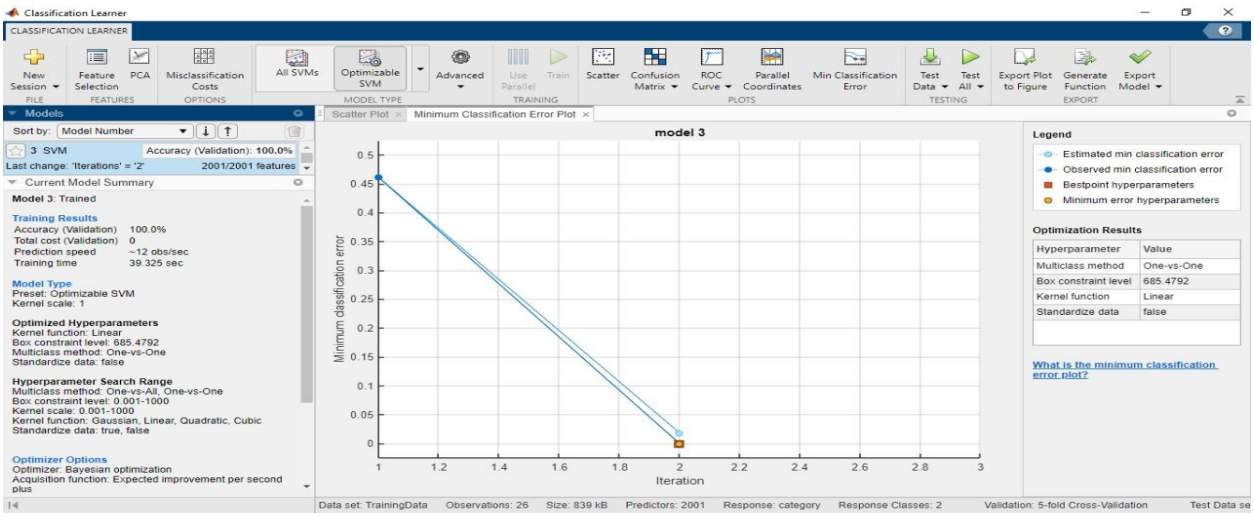

Figure 5: SVM validation details and classification error plot

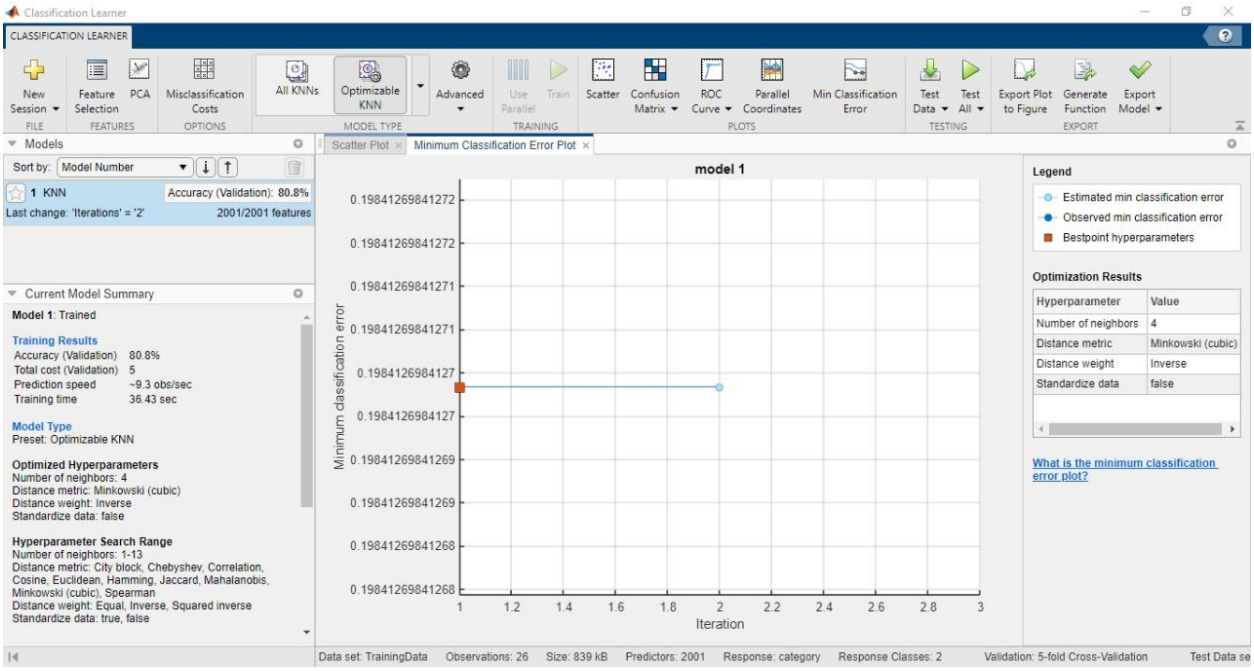

Figure 6: KNN validation details and classification error plot

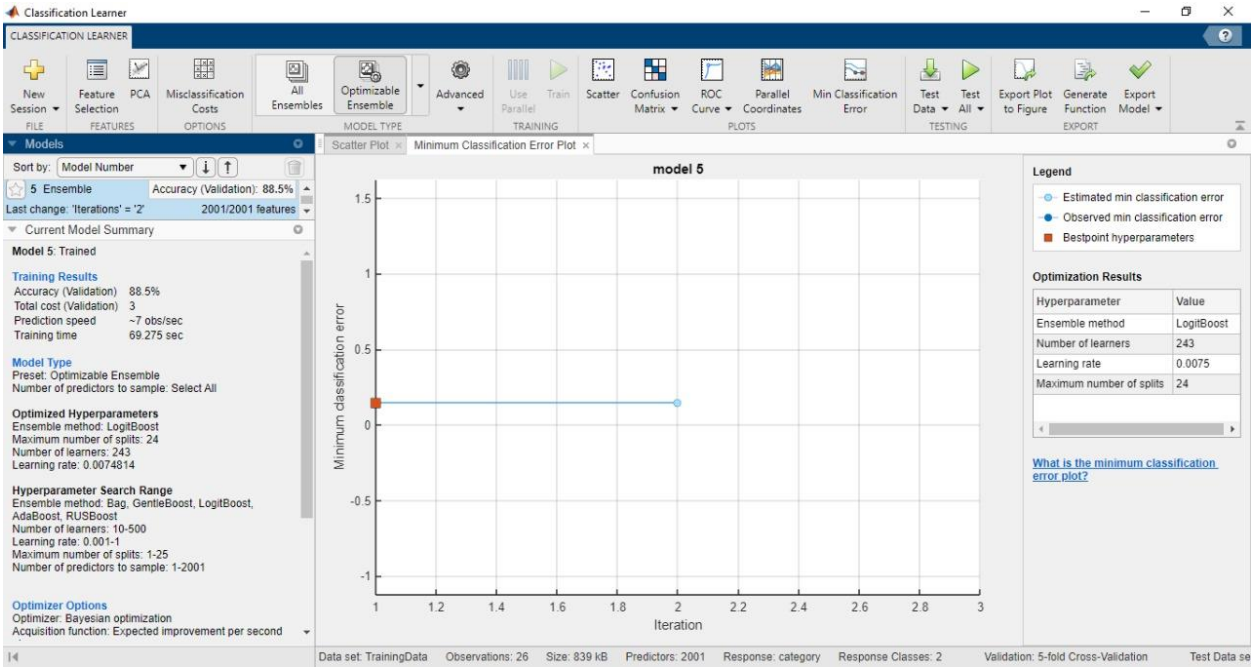

Figure 7: Ensemble validation details and classification error plot

Confusion matrices

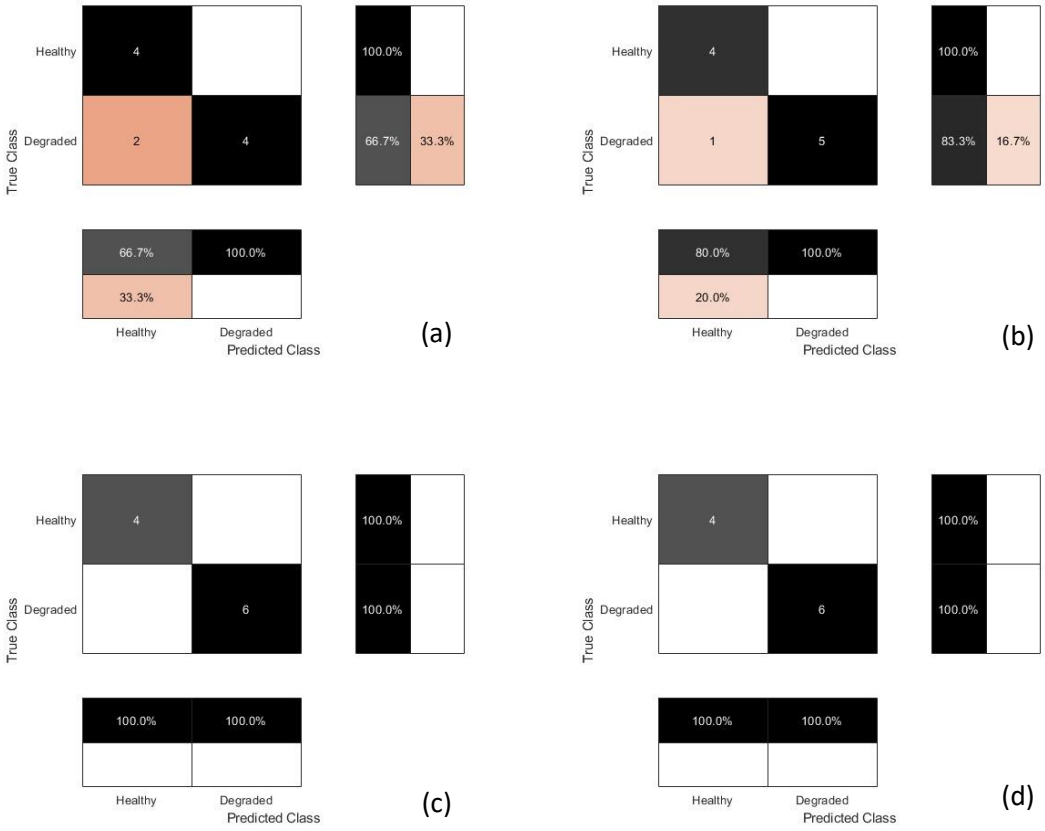

Figure 8: Confusion matrices of (a) KNN, (b) Ensemble, (c) SVM, and (d) ANN

The validation and testing accuracies of the four models are summarized in Table 2. The results show that ANN, Ensemble, and KNN gave good accuracies in the validation stage in this particular order. For testing, KNN had the same accuracy, while ANN and Ensemble models showed an increased accuracy for the testing data classification. The SVM model shows 100 % accuracy for both validation and testing data classification which can be attributed to model overfitting. The confusion matrices in Figure 8 elaborate the table results of testing accuracy. The KNN model (Figure 8 (a)) predicted 4 out of 4 healthy, 4 out of 6 degraded, and 2 were predicted as healthy while they were degraded. The Ensemble model (Figure 8 (b)) predicted 4 out of 4 healthy, 5 out of 6 degraded, and 1 was predicted as healthy while it was degraded. The accuracy can be calculated by the following formula:

(TP+TN)/(TP+TN+FP+FN), where,

TP= True Positive (healthy)

TN= True Negative (degraded)

FP=False positive (number of incorrectly predicted positive cases)

FN=False negative (number of incorrectly predicted negative cases)

current use

Business Case:

Brain Computer Interface (BCI) is a technology that enables direct communication between the brain and an external device, such as a computer or a prosthetic limb, without the need for any muscular activity. BCI systems typically involve using brain signals, often recorded through methods like EEG or fMRI (functional magnetic resonance imaging), to control external devices or to provide feedback to the user about their brain activity. BCI technology holds promise for a variety of applications, including assistive technology for people with disabilities, neurofeedback training, and even enhancing human-computer interaction in gaming and other fields.

In the automotive industry, BCI is used to develop driver assistance systems that monitor the driver's cognitive state and alertness, providing warnings or interventions to prevent accidents caused by fatigue or distraction. When using EEG-based BCI, it is important to determine the quality of EEG signals as degraded signals with more than normal artifacts can give compromised results.