BS6200 Project Report: Machine Learning-based EEG siganl classification

Yaoding Chen

ychen096@e.ntu.edu.sg

Nanyang Technological University, Singapore

1 Introduction

Epilepsy is a neurological disease characterised by recurrent, unprovoked seizures. It is one of the most common neurological disease, affecting 70 million worldwide (Paul, 2018). About 30% of people with epilepsy are medically intractable, which is associated with adverse outcomes, such as serious comorbidities, injury and death (Kuhlmann et al., 2018). Clinically, seizure can be recorded in electroencephalogram (EEG), which is an electrophysiological monitoring method that measures brain electrical activity. Since the disease onset is considered to be potentially dangerous and largely unpredictable, close monitoring of EEG signals for early detection of seizure is important to avoiding severe adverse outcomes caused by the disease. Thus, developing an automated approach for seizure detection based on machine learning has drawn great interest in the field of both biomedical sciences and data science.

Technically, EEG signals measures voltage fluctuations due to intra-neuron ionic current, which can be caused by seizure and a variety of non-seizure states, such as eye open and eye close with different patterns. Distinguishing the signals from each other can improve our understanding of brain function with the different patterns identified to be involved in the signals, which will increases our knowledge of neuropathological processes, eventually contributing to better seizure detection, prediction and treatment. Thus, lately various machine learning approaches have been proposed to decode the signals. In this paper, we will construct and assess a series of supervised learning approaches, along with different feature engineering approaches, to classify EEG signals into 5 different classes.

2 Problem Statement

The project aims to apply and develop suitable machine learning approaches to perform multi-class classification of EEG signals with a simplified, machine learning-ready version of EEG dataset from the University of Bonn (Andrzejak et al., 2001), which is acquired through UCI Machine Learning Repository (Asuncion & Newman, 2007), to order to compare and contrast how different data preprocessing approach may affect choices and accuracy of different machine learning models. Thus, we propose hypotheses as follows:

1. Different machine learning approaches can classify EEG signals into 2+ classes with good accuracy.
2. Dimension reduction methods such as principal component analysis (PCA) can retain the accuracy of machine learning models despite reduced size of input data
3. Feature engineering methods based on wave analysis can increase the accuracy of machine learning classifiers.

Our ultimate aim is develop optimised models for the multi-class classification of EEG signals and compare and analyse their results with insights into model choices.

3 Dataset Description

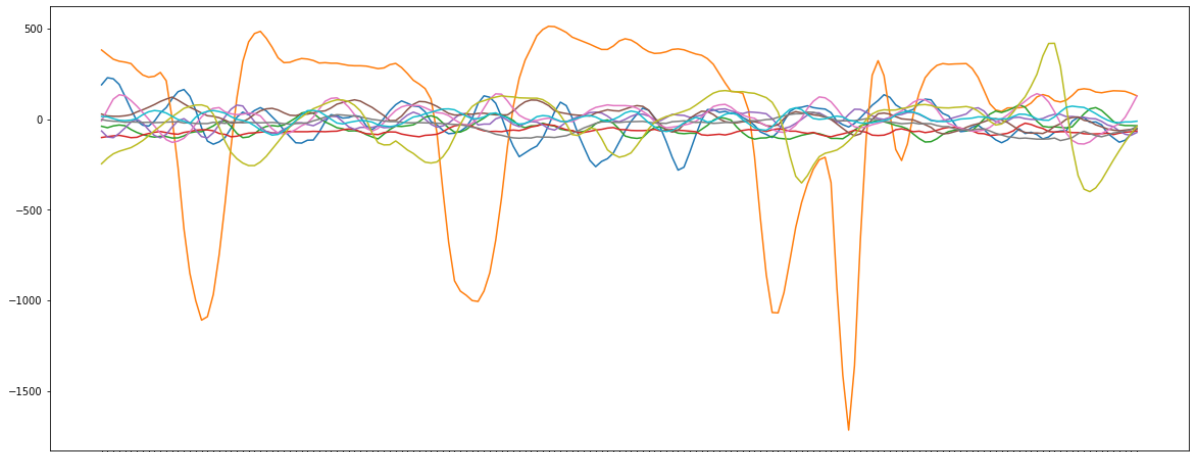


Figure 1. Example EEG signals

As illustrated in Figure 1, EEG signals are time-series data involving fluctuations and waves. According to the description on UCI Machine Learning Repository, the whole dataset consists

of 11500 pieces of EEG signals data from 500 individuals. Each piece of data measures a time period of 23.6 seconds with 178 data points. The dataset classify all EEG data into five distinct types by biological significance:

1. during seizure activity
2. in brain tumour sites
3. in healthy brain
4. during eye close
5. during eye open

Among the data classes, only Class 1 is considered to have seizure activity, while the rest are non-seizure but related to other defined biological states. As illustrated in Figure 2, recordings of seizure activity have much more volatile curves than other classes of recordings. Thus, a good number of works have been aimed to perform binary classification of the signals, which can achieve high accuracy (Resque et al., 2019).

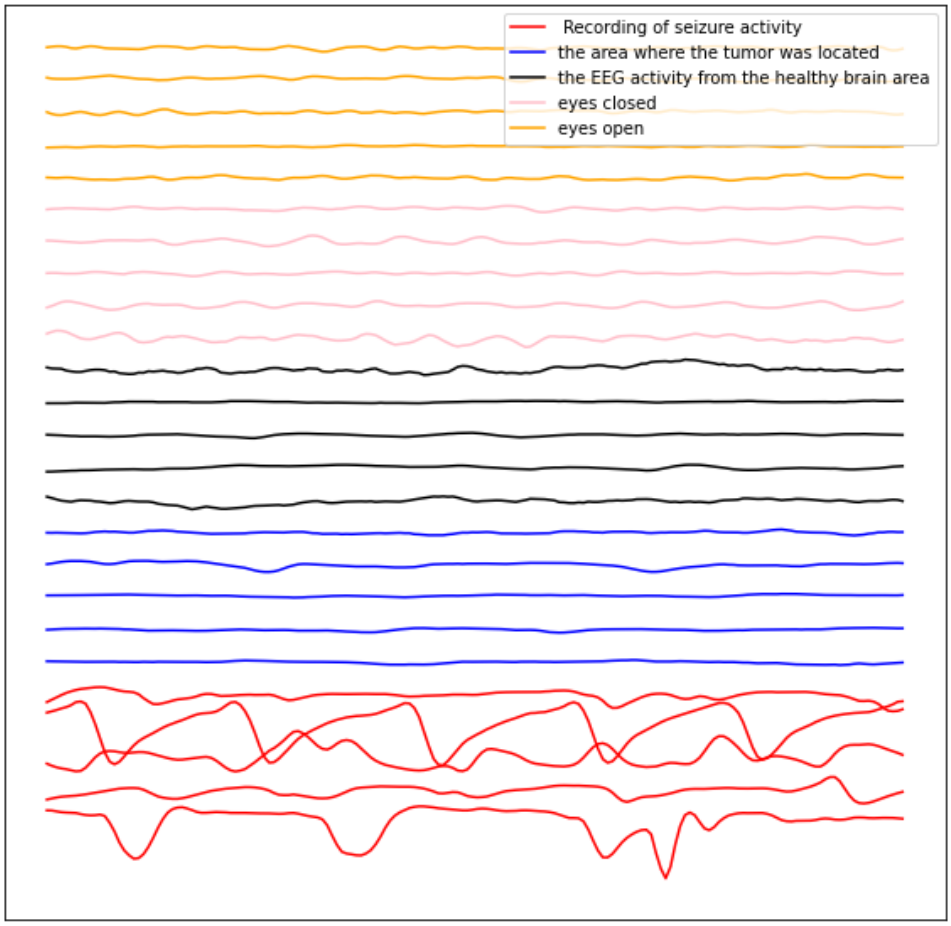
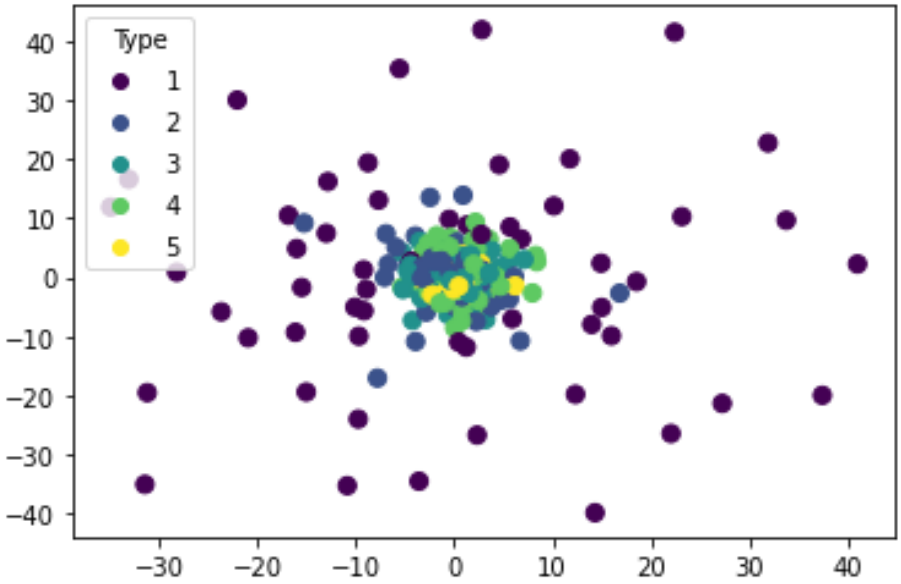
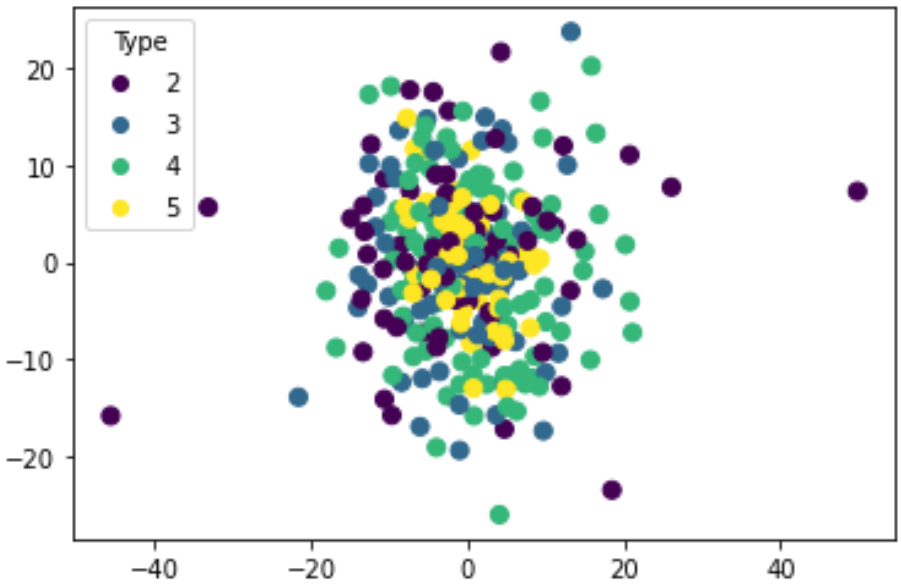


Figure 2. Classification of EEG signals

To further visualise the differences between data classes, we leverage multidimensional scaling (MDS) to visualise the high dimensional data of EEG (Cox & Cox, 2008), as shown in Figure 3. In Figure 3a, it is all non-seizure cases cluster in the centre, whereas the rest sparsely located off the central cluster are mostly seizure cases. In Figure 3b, we further aims to visualise high-dimensional non-seizure data with MDS, where different classes of data tend to mix with each other.



3a) visualisation of all classes of EEG data (Class 1-5)



3b) visualisation of non-seizure classes (Class 2-5)

Figure 3. Visualisation of different types of EEG signal clustering using MDS

Thus, the cluster suggests relatively higher difficulty of 5-class classification when compared with binary classification, where most models used can reach accuracy as high as more than 90%. Also, for binary classification, the chance of getting accurate results are 50%, which suggests a much higher baseline than 20%-baseline in the 5-class classification question. Thus, any model with > 20% accuracy outperforms random guess. In this model, we will validate whether most machine learning techniques can outperform this and further compare the models.

4 Data Preprocessing

4.1 Raw Data

4.1.1 Feature Scaling

Our first step is feature scaling to limit the range of the data, which gives us certain benefits. First, feature scaling can avoid feature variance from affecting the performance of models such as support vector classifier (SVC) and K-nearest neighbours (KNN) classifier that are prone to unequal variance. Second, it is known that data normalisation is known to improve performance in SVC (Liu, 2011) and other models. Third, for models not sensitive to data distribution, deploying the same data scaling method also makes it possible to compare between machine learning models using the same set of features. Lastly, it is also important to our next step, principal component analysis (PCA), as it tries to get the data with largest variance.

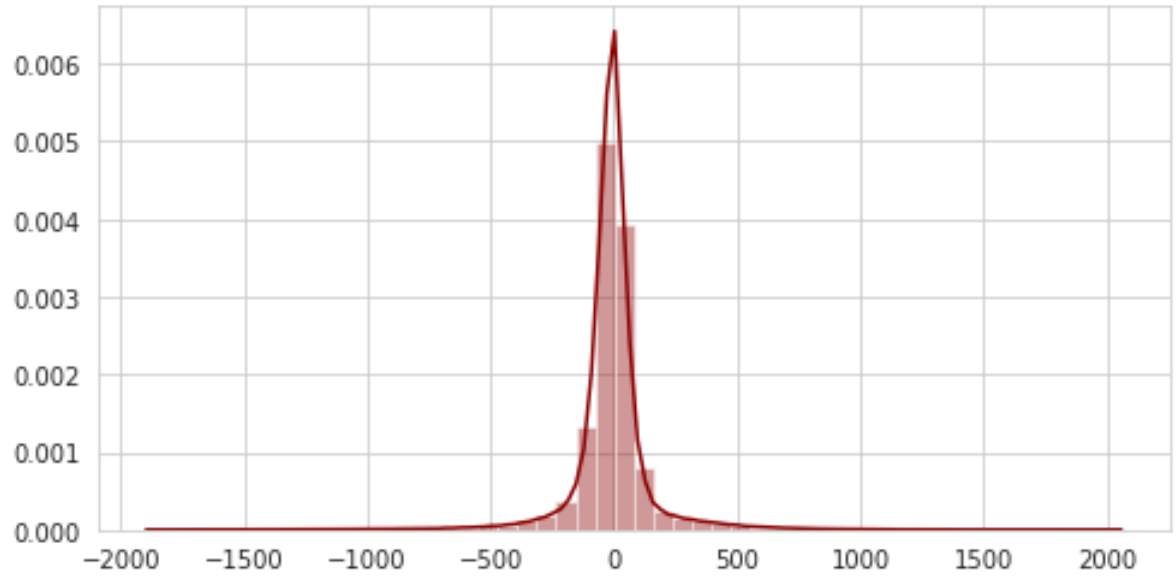


Figure 4. Distribution plot of EEG data

Thus, to choose a proper data scaling technique, we plot the data distribution in Figure 4, which the EEG data is already normally distributed. Thus, we standardise the data with StandardScaler from the scikit-learn package to create a standard normal distribution of the data and to scale the data within [0,1] for models with raw input data. Since our engineered features have quite different ranges, to normalise them, we use *normalize* function from the sckit-learn package to ensure each data have the same range and normally distributed.

4.1.2 Dimension reduction with principal component analysis (PCA)

As is mentioned above, PCA is a technique used for dimension reduction by identifying components with largest variance. For our data with 178 dimensions in the input, with PCA can be beneficial in several aspects. First, it can remove correlated features in the dataset, which simplifies the input dimensions. Second, with less features, the speed of algorithm execution is faster. Third, it is likely that PCA can reduces overfitting when often occurs when there are too many variables, which may contribute to better model performance. Here we plot the scree plot of our PCA result as illustrated in Figure 5. To get 95% of the explained variance ratio, we select 39 principal components with largest variance.

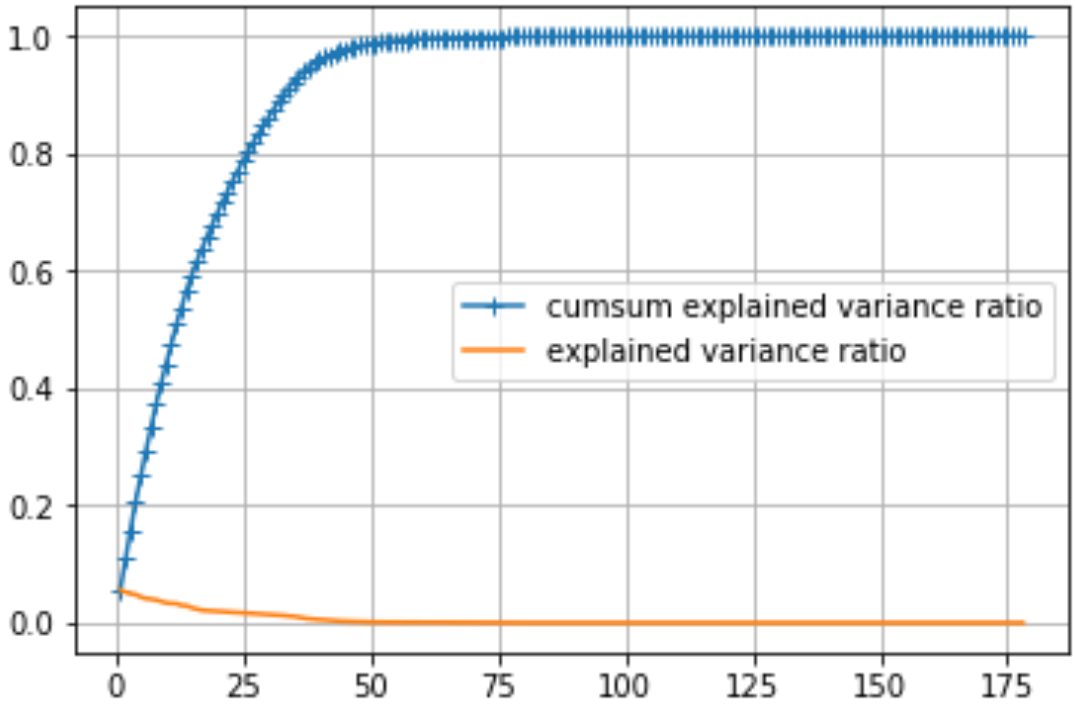


Figure 5. PCA scree plot

4.2 Feature Engineering and Selection

4.2.1 Fast Furrier transformation (FFT)

Although it is possible for machine learning models to learn the features in the dataset with relatively good accuracy, we also try to dissect the EEG signals based on existing knowledge of EEG analysis, which may help us to build more interpretable models. EEG signals are typically separated by 4 pre-defined frequency bands, namely alpha, beta, gamma, theta and delta waves, which are considered to be correlated with different spectra of neurological diseases (Newson & Thiagarajan, 2019). FFT is an algorithm to efficiently calculate discrete Fourier transform (DFT), which converts equally-spaced samples of a function into a same-length sequence of a complex-valued function of frequency, namely the discrete-time Fourier transform (DTFT). Therefore, FFT allows interpretation of the data in a frequency domain rather than a time domain, which is typically used in biomedical signal processing (Krishnan & Athavale, 2018). In this way, we can analyse the data of different frequency bands.

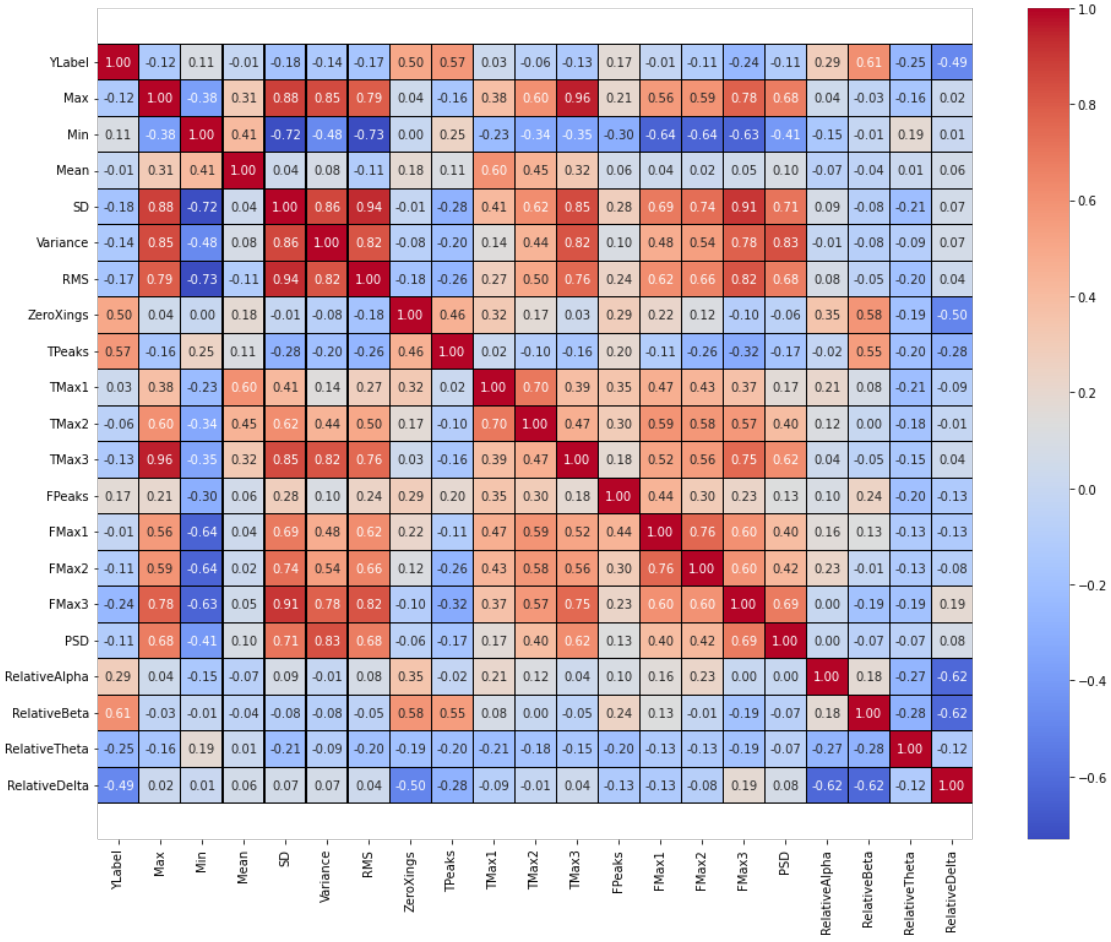


Figure 6. Heatmap of engineered features

In our project, we engineer a series of variables based on FFT, which can be illustrated in the heatmap below (Figure 6). The feature is constructed based in a previous work and analyse a variety of wave features of the signals. These features are describes as follows:

- Max, Min, Mean, SD, Variance, RMS, ZeroXings refer to the max, min, mean, standard deviation, variance, root mean square, zero crossings of 178 data points in each piece of the data respectively.
- TPeaks refers to the number of peaks in the time domain, while TMax1, TMax2, TMax3 refer to the respective values of three largest peaks.
- FPeaks refers to the number of peaks in the frequency domain, while FMax1, FMax2, FMax3 refer to the respective amplitudes of three largest peaks.
- PSD is the power spectral density of the 178 data points in each piece of the data respectively.
- RelativeAlpha, RelativeBeta, RelativeTheta, RelativeDelta refer to the relative distribution of the alpha, beta, theta and delta wavebands.

With the features constructed, we further apply Lasso regression analysis to select the features of importance to classification. We use grid search to tune alpha in Lasso. As a result, 10 variables, i.e., Max, Min, ZeroXings, TPeaks, TMax1, TMax2, FMax1, RelativeAlpha, RelativeBeta and RelativeTheta are selected as the analysis result shows the value of importance larger than 0.

5 Model Choices

To make 5-class classification, we use 4 classical machine learning models, 2 ensemble learning models, and 1 deep learning model with scikit-learn in Python (Pedregosa et al., 2011). We aim to test them in three different set of features to build raw input-based, PCA-based and engineered feature-based models.

In a number of classifiers, multiclass classification is often transformed into one-vs-one or one-vs-rest classification, which we will provide more details about later in this section. For tuning the hyperparameter, we use grid search with cross validation. Here we aim to describe the models and how they deal with multiclass classification and how we tune the models as follows.

5.1 Logistic Regression (LR)

Logistic regression is designed for modelling the probability of certain binary classification. Here, to deal with multiclass classification, the one-vs-rest scheme is adopted. Thus, five independent logistic regression models are trained to perform the classification. For each model, we aims to fit the probability p and an n-array of $(x_1, x_1, x_1, \dots, x_n)$ into the following equation:

$$log\frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

To tune the model, we use grid search to find the best β_0 for the model. In raw input-based model, the best value of β_0 is found to be 0.07896522868499725. In PCA-based model, it is 0.18047217668271703.

5.2 K Nearest Neighbour (KNN)

K nearest neighbour algorithm classifies data according to majority voting or averaging of distance between the data point and different clusters. In scikit-learn, majority voting is by default and multiclass classification is natively supported. The number of nearest neighbours are tuned with grid search, which consistently turns out to be 2 in both raw input- and PCA-based models.

5.3 Support Vector Classifier (SVC)

SVC calculates hyperplane that separate the data points with largest margin in a high-dimension space, where the flexibility of kernel functions help make good classification for linear and non-linear data. Though SVC is designed for binary classification, to address multiclass classification, the sckit-learn implementation of SVC uses a one-vs-one scheme, which generates N* (N-1)/2 binary classifier models given a N-class classification question.

We tune our SVC model for optimising the degree of the polynomial kernel function to better fit the data and the regularisation parameter C to avoid overfitting. In all models we train, the optimal degree number is 3, whereas C equates 100, 10000 and 100 in raw input-based, PCA-based, and FFT-based models respectively.

5.4 Naïve Bayes Classifier (NBC)

NBC is based on the Bayes theorem, which allows us to calculate *a posteriori* probability based *on a priori* probability, hence predicting the chance of variables of interest based on the data found on the hypothesis and the total data, which then classify the data according to larger *a posteriori* probability. Thus,

$$posterior = \frac{prior \times likelihood}{evidence}$$

Thus, the equation can expressed mathematically:

$$p\,(C_k \mid X) \propto p(C_k, X) = p(C_k) \cdot \prod p(x_n)$$

In natural language processing, NBC, which natively support multiclass classification, is often used for predict word tags, which treat different words as independent events to calculate and compare the chance of different tags. However, assumption of independence of the variables may not hold in our case due to multilinearity and noise, yet we still try the model in our dataset, considering its popularity as a base estimator for comparisons. We do not perform grid search for parameters of the model. Instead, the input dataset is divided by 7:3 for train/test datasets for the classifier.

5.5 Random Forest Classifier (RFC)

RFC is an ensemble learning method which fit fits a number of decision tree classifiers on a variety sub-samples of the dataset which then uses averaging to improve classification accuracy. Decision tree learning is a non-parametric supervised learning which learns features with a set of simple decision rules, which is called leaves. However, the approach often causes overfitting, which makes pruning of leaves highly important.

Here in RFC, each tree is the ensemble built from a bootstrap sample from the training set. The best split of nodes are found either from all input features or a random subset of size. By calculating average prediction results of individual decision trees, RFC is able to reduce some error caused by overfitted trees. In scikit-learn, RFC calculates average probabilistic prediction rather than use a majority-voting scheme.

To tune the model, we apply grid search to the maximum depth of the tree (*max_depth*) and the number of features to consider when looking for the best split (*max_features*) to control the size of each tree. Thus, optimally, the parameters equate 9 and 33 in the raw input-based model respectively, and 9 and 2 in the PCA-based model.

5.6 Gradient Boosting Classifier (GBDT)

Gradient boosting classifier, or gradient boosting decision trees (GBDT) uses many weak learning models together, typically decision tree classifiers. Thus, the model can be illustrated as an additive model with a weighted sum of a set of base learners $f(x, \theta_m)$ shown below:

$$F(x; \{a_m, \theta_m\}) = \sum_{m=1}^{M} a_m f(x, \theta_m)$$

With addition of new learners to the classifier, old trees are not adjusted, yet a gradient descent procedure is used to find the optimal a_m that minimises the average value of the loss function θ_m when adding the new tree. Although arbitrary differentiable loss functions can be used, deviance is by default in scikit-learn. Multiclass classification is natively supported as the labels do not usually affect calculation of the loss function.

To tune the model, we apply grid search to the maximum depth of the individual regression estimators (*max_depth*) and the number of features to consider when looking for the best split (*max_feature*). Thus, optimally, in raw input and PCA based models, *max_feature* is the square of the number of the features and *max_depth* is 9, while in the engineered feature based model, *max_feature* is the logarithm of the number of the features to base 2 and *max_depth* is 7.

5.7 Multilayer Perceptron (MLP)

MLP is a class of feedforward artificial neural network, which can extract features itself with hidden layers, which are helpful to learn high dimensional, multi-class data. In each layer, there is a set of extracted features, called nodes, which are extracted from the prior layer. All the nodes in the same will be used to calculate the nodes in the next layer with different weights. The weights are adjusted with backpropagation to minimise the loss function, which in scikit-learn is the log-loss function by default in a gradient descent manner. To tune the model, we

apply grid search to the number of hidden layers, which is optimally 512 in raw input and PCA based models and 256 in the engineered feature based model.

5.8 Majority Voting Classifier (MVC)

We also try to combine our best models with MVC, which is an ensemble learning scheme where the majority of the predicted class labels from different base learners is the output class label of the ensemble learner. This is thought to balance out the individual weaknesses of individual models, thus increasing classification accuracy.

6 Classification with raw data

6.1 Training and testing procedures

Here we use a variety of models which include LR, SVC, NBC, MLP, RFC, GBDT for processing the data without feature engineering. We first take a grid search approach with cross validation to tune hyperparameters of our models. With the tuned parameters, we further divide the data into 70% training and 30% testing and then re-run the model for training and testing, which produces the model accuracy in the raw dataset and PCA-based dataset.

6.2 Model performance

As mentioned in the Data Preprocessing section, PCA increases training results to different extent, as discussed in the Data Preprocessing section. A summary of training results is presented in Table 1.

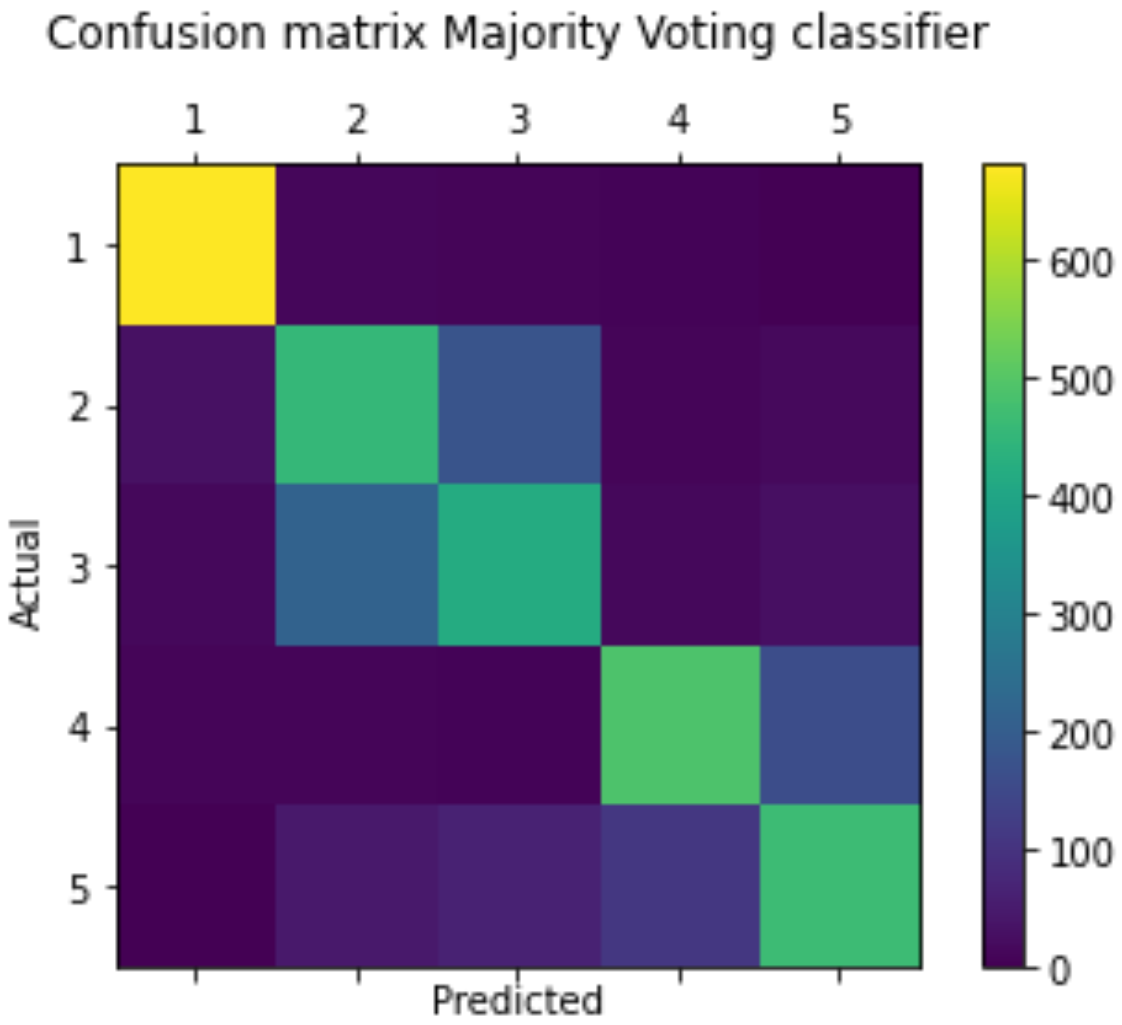
Models	Accuracy (raw input)	Accuracy (PCA-based)	Change
LR	0.25645	0.25732	0%
SVC	0.68792	0.69922	1%
KNN	0.49406	0.51898	2%
NBC	0.44538	0.62967	18%
MLP	0.69284	0.71544	2%
RFC	0.51463	0.62417	11%
GBDT	0.68937	0.72675	4%

Table 1. Accuracy change of models before and after PCA

Although all models are better classifiers than guessing, logistic regression always have the worst performance, which may suggest nonlinear nature of the categorical data which may be unsuitable for working with logistic regression. KNN can correctly classify around half of the result. However, the distance-based method may be subject to the mixed-up data points between clusters, thus contributing to a lower performance. Interestingly, NBC and RFC get similar accuracy at around 62%. Here removal of correlated variables may partly explain the increased accuracy in naïve Bayes classifier after PCA. It is notable that SVC, MLP, and GBDT can make classification as accuracy as around 70%.

6.3 Majority Voting Classifier

Thus, to leverage the 3 highly accurate models (SVM, MLP and GBDT), we further apply them to a majority voting classifier, which produces an accuracy of 73%, outperforming the three models together with the PCA-based raw input dataset, as illustrated in Figure 7.



	precision	recall	f1-score	support
1	0.93	0.96	0.94	709
2	0.62	0.66	0.64	685
3	0.62	0.61	0.61	690
4	0.78	0.73	0.75	676
5	0.69	0.68	0.68	691
accuracy			0.73	3451
macro avg	0.73	0.73	0.73	3451
weighted avg	0.73	0.73	0.73	3451

Figure 7. Classification report of majority voting classifier (raw input + PCA)

As expected, Class 1 has very high precision, recall and F1 score, while the rest have much lower classification accuracy, which pulls down the overall performance of the classifier. Similar issues should happen in all the above-mentioned models, which is however yet to examined in further studies.

7 Classification with engineered features

To test whether models based on engineered features can outperform those based on PCA or raw input, we train three models with grid search for hyperparameter tuning. The three models are SVC, MLP and GBDT. We use the same approach to train and test the models yet based on a engineered feature dataset scaled with normalisation. Table 2 and 3 show the summary of the model performance:

Models	Accuracy (Raw + PCA)	Accuracy (FFT)	Accuracy (FFT + Lasso)	Change (FFT vs. FFT +Lasso)
SVC	0.69922	0.72012	0.71429	-1%
MLP	0.71544	0.68870	0.68609	0%
GBDT	0.72675	0.73118	0.71416	-2%

Table 3. Comparison of accuracy between FFT-based and PCA-based models

It is clear that despite significantly reduced input data, the models have similar performance or outperforms those using PCA-processed or raw input data, which suggests that the constructed

features are able to capture most of the features that are important to classification. However, it is interesting that feature selection using Lasso reduces the accuracy of classification. As we have mentioned correlated parameters previously and as is shown in Figure 6, there is a good number of correlated variables, whereas Lasso tends to drop one of the features, leading to loss of information. This is quite different from PCA, which calculates a new variable based on correlated variables, thus preserving more information.

8 Conclusions and future works

In this project, we apply a number of popular machine learning methods for a 5-class classification question based on seizure detection dataset. First, we show that all the models can classify the data with better accuracy than random guess, yet we are still missing certain popular classifiers such as recurrent neural networks and convolutional neural networks. Yet, although we show that PCA can increase the classification accuracy in raw input dataset, it is unclear whether this holds true to human engineered features or whether it can do better than Lasso-based feature selection. Second, there are still a wide range of other human engineered features such as moving average that is worth constructing and testing in addition to our engineered features, which may help us to construct better, more interpretable models. Finally, a lasting question in the field is whether seizure is predictable. Although some proposed methods such GenericPred (Golestani & Gras, 2014) to predict seizure in a short time period, we have not got enough time to deploy the model to our dataset, which is a good future direction for our next step.

9 Code Availability

The code for reproducing the results are available in Jupyter Notebook format at https://github.com/ydchen17/SeizureDetection. The dataset is supplied within the notebooks. Further changes, if applicable, may be made to the GitHub repository.

10 References

Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., & Elger, C. E. (2001). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, *64*(6), 61907. https://doi.org/10.1103/PhysRevE.64.061907

Asuncion, A., & Newman, D. (2007). *UCI machine learning repository*. Irvine, CA, USA.

Cox, M. A. A., & Cox, T. F. (2008). Multidimensional scaling. In *Handbook of data visualization* (pp. 315–347). Springer.

Golestani, A., & Gras, R. (2014). Can we predict the unpredictable? *Scientific Reports*, *4*(1), 6834. https://doi.org/10.1038/srep06834

Krishnan, S., & Athavale, Y. (2018). Trends in biomedical signal feature extraction. *Biomedical Signal Processing and Control*, *43*, 41–63. https://doi.org/https://doi.org/10.1016/j.bspc.2018.02.008

Kuhlmann, L., Lehnertz, K., Richardson, M. P., Schelter, B., & Zaveri, H. P. (2018). Seizure prediction — ready for a new era. *Nature Reviews Neurology*, *14*(10), 618–630. https://doi.org/10.1038/s41582-018-0055-2

Liu, Z. (2011). A method of SVM with normalization in intrusion detection. *Procedia Environmental Sciences*, *11*, 256–262.

Newson, J. J., & Thiagarajan, T. C. (2019). EEG Frequency Bands in Psychiatric Disorders: A Review of Resting State Studies . In *Frontiers in Human Neuroscience* (Vol. 12, p.

521).

Paul, Y. (2018). Various epileptic seizure detection techniques using biomedical signals: a review. *Brain Informatics*, *5*(2), 6. https://doi.org/10.1186/s40708-018-0084-z

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Resque, P., Barros, A., Rosário, D., & Cerqueira, E. (2019). An Investigation of Different Machine Learning Approaches for Epileptic Seizure Detection. *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, 301–306. https://doi.org/10.1109/IWCMC.2019.8766652