Predicting the development of central neuralgia in patients with spinal cord injury using EEG data: an exploration of feature engineering and machine learning methods

Ruixian Zhang

Junnan Zheng

Zhexi Ju

Zhenghao Lin

Min Ma

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8QQ

A project report presented in part fulfilment of the requirements of the

Degree of Master of Science at The University of Glasgow

01/12/2023

Abstract

This report focuses on the use of electroencephalography (EEG) data to predict the development of central neuropathic pain (CNP) in patients with spinal cord injury (SCI). Given that approximately 50% of SCI patients will develop CNP, the purpose of this study was to analyse EEG data to identify in advance those patients who are at higher risk of developing CNP. The study employed advanced feature engineering and machine learning techniques to process and analyse high-dimensional EEG data from a small sample of SCI patients.

We first performed a thorough preprocessing of the EEG data, including data cleaning, normalisation, and dimensionality reduction. Subsequently, various feature selection and extraction methods were applied to optimise the dataset and improve the predictive power of the model. The main machine learning models used in the research include list one or two mainly used models, such as support vector machines, random forests, etc.

The results show that features related to CNP development can be effectively identified from EEG data through sophisticated feature engineering and machine learning methods. Our model successfully distinguished patients who developed CNP from those who did not, demonstrating high accuracy and reliability. This discovery provides new possibilities for early diagnosis and intervention, helping to improve long-term treatment outcomes for SCI patients.

This study not only highlights the potential of machine learning for medical data analysis, but also provides valuable insights into the future use of EEG data more broadly in neuroscience and clinical practice.

Contents

Chapter 1: Introduction

1.1 Introduction

Central neuralgia (CNP) is a chronic pain condition common in patients with spinal cord injury (SCI), characterised by a painful response to non-painful stimuli, manifested by electric shock-like pain, pins and needles, and numbness. It is estimated that about 50% of SCI patients will develop CNP, which not only seriously affects the patients' quality of life, but also its management and treatment has been a challenge in neuroscience and clinical medicine due to the lack of effective treatments.

The main motivation of this study was to explore the use of electroencephalogram (EEG) data as a biomarker to predict whether patients with SCI will develop CNP. EEG, a non-invasive and relatively economical technology capable of recording the brain's electrical activity, may contain critical information about the development of CNP. Currently, determining whether a patient with SCI will develop CNP mainly relies on subjective clinical assessment, which is time-consuming and error-prone. Therefore, developing an objective prediction model based on EEG data has important clinical value.

This report aims to analyse EEG data of SCI patients by applying advanced machine learning techniques and feature engineering methods. Our goal is to identify EEG features that can effectively predict the development of CNP and build an efficient classification model. In this way, we hope to support early diagnosis and intervention for SCI patients, thereby improving their long-term treatment outcomes and quality of life.

In the following chapters, we will detail the methods used for feature engineering, demonstrate the constructed machine learning model, and analyse and discuss the performance and clinical application potential of the model.

1.2 Aims

1. Data collection and preprocessing

- Electroencephalogram (EEG) data were collected from patients with spinal cord injury.

- Perform necessary data pre-processing, including de-noising, standardisation, and data cleaning, to ensure data quality and analyzability

2. Feature Engineer

- Apply feature engineering techniques, such as feature selection and extraction, to identify EEG data features relevant to CNP development.
- Analyse and determine the EEG signal parameters that most effectively represent CNP characteristics.

3. Model development and validation

- A model was constructed using machine learning algorithms to predict the likelihood of SCI patients developing CNP.
- The accuracy and generalisation ability of the model were verified by cross-validation and other statistical methods

4. Result analysis and interpretation

- Analyse the predictive results of the model and identify key predictive indicators and patterns.
- Explain the association between EEG data features and CNP development, and its potential application in clinical prediction.

5. Clinical application exploration

- Explore the possibility of applying research findings to clinical practice, including early diagnosis, personalised treatment plans, and intervention strategies.
- Based on the research results, suggestions for future clinical practice and research direction are put forward.

Chapter 2: Background

Central neuralgia (CNP) is a common chronic pain condition after spinal cord injury (SCI), affecting the quality of life and mental health of SCI patients. CNP is characterised by an abnormal pain response to non-painful stimuli, often manifesting as electric shock-like pain, pins and needles, or numbness. Currently, understanding of CNP remains limited, and treatments are inconsistently effective and often associated with side effects.

Electroencephalography (EEG), as a technique for recording brain electrical activity, has been widely used in neuroscience research and clinical practice. The advantages of EEG are its non-invasiveness, real-time monitoring capabilities, and relatively low cost. In recent years, with the development of machine learning technology, EEG data analysis has shown new potential in neuropathology, cognitive science, and clinical diagnosis. Especially in predicting and classifying neurological diseases, EEG combined with machine learning technology has made some breakthroughs.

Despite this, the potential of EEG data has not yet been fully exploited in CNP research. Current prediction of CNP after SCI mainly relies on clinical assessment, and these methods usually rely on patients' subjective reports and doctors' empirical judgments. This approach is not only inefficient but also susceptible to personal bias. Therefore, it is particularly important to develop a CNP prediction model based on objective biomarkers.

This study focuses on using EEG data combined with advanced feature engineering and machine learning techniques to predict whether SCI patients will develop CNP. Our goal is to build a model that can efficiently and accurately predict the development of CNP by analyzing specific patterns and characteristics in EEG data. This not only provides earlier intervention opportunities for SCI patients, but also provides a new perspective for the research and treatment of CNP.

By deeply exploring the potential of EEG data, this study aims to provide a new scientific basis for understanding and predicting CNP, while also bringing new enlightenment to neuroscience research and clinical practice.

Chapter 3: Methods

3.1. Chi-Square Test

The Chi-square test is a statistical method used to assess the independence of two categorical variables. It operates by comparing the actual counts in each category to the expected counts that would occur if the variables were independent. This test computes the Chi-square statistic, denoted as χ^2, which represents the sum of the squared differences between observed and expected frequencies, each divided by the expected frequency. This approach allows for a quantifiable measure of how much the observed data deviates from what would be expected under the assumption of independence between the variables (sklearn, 2023).

The Chi-square test is highly effective for analysing categorical data, but its effectiveness is limited to such data types and relies on having a sufficiently large sample size for accurate results. In cases of small datasets or when dealing with a large number of categories, the test may not be as effective in identifying relationships. This is because the accuracy of the Chi-square test diminishes with smaller sample sizes, and a high number of categories can lead to an increased chance of error or misinterpretation of the data (S.M. Gajawada, 2019).

3.2. Pearson Correlation

The Pearson Correlation Coefficient, symbolised as r, is a statistical metric that quantifies the strength and direction of a linear relationship between two continuous variables. It ranges between -1 and +1, where -1 indicates a perfect negative linear correlation, +1 signifies a perfect positive linear correlation, and 0 denotes the absence of any linear relationship. In the context of feature selection, the Pearson Correlation is often utilised to eliminate features that do not exhibit a linear relationship with the target variable.

The Pearson Correlation is tailored to identify linear relationships and may not accurately reflect nonlinear interactions. Furthermore, the Pearson Correlation can be influenced by outliers, as extreme values can disproportionately affect the results. It's also predicated on the assumption that the data is normally distributed, and deviations from this assumption can impact the accuracy and reliability of the correlation coefficient (R. V, 2022).

3.3. Distance Correlation

Distance correlation is a statistical method that evaluates both linear and nonlinear associations between variables. Unlike methods that rely on ranks, distance correlation is based on the actual distances between data points in the data space, considering their actual values (Tan et. al., 2022). This approach allows it to capture both linear and non-linear associations, providing a more comprehensive analysis compared to the Pearson correlation.

However, there are notable considerations with distance correlation. It is computationally intensive, particularly for large datasets, which can be a limitation in certain applications. Additionally, interpreting the correlation values derived from distance correlation can be less intuitive than those obtained from Pearson's correlation, potentially posing challenges in communicating and understanding the results (Das, R., Kasieczka, G., & Shih, D., 2022).

3.4. Forward Selection

Forward selection is a methodical approach in feature selection that begins with a null model, which contains no predictors or features. In this process, the algorithm sequentially considers adding each feature that is not currently in the model. At each step, it evaluates and adds the feature that most significantly improves the model based on a specific criterion. The procedure concludes when the inclusion of additional features does not yield a significant improvement in the model. The stopping point can be determined by various factors, such as reaching a predetermined number of features, achieving a specific threshold for the improvement metric, or arriving at a point where no additional variables significantly enhance the model (Mayo, M. 2018).

Forward selection is a straightforward and intuitive method for model building, appealing for its step-by-step approach that is easy to implement and interpret. This method is versatile, applicable across various machine learning models, not limited to linear regression. Particularly in datasets with many features, forward selection is computationally more efficient than techniques requiring the fitting of all possible feature combinations. However, its path-dependent nature means that the final model can be influenced by the order in which variables are entered, potentially leading to suboptimal outcomes if an early choice precludes a better variable added later. Forward selection may not effectively capture complex relationships, such as interactions or non-linearities (Singh, H., 2021).

3.5. Backward Elimination

Unlike forward selection, backward elimination starts with a comprehensive model that includes all potential predictor variables. In this process, the algorithm evaluates the impact of removing each feature one by one. The chosen feature for removal is typically the one whose absence least degrades the model's performance or, alternatively, the one with the lowest statistical significance. This procedure is iterative, involving the removal of one feature at a time, with the model being re-evaluated after each elimination. The process ceases when the removal of additional features substantially worsens the model's performance, with the stopping criteria often being a specific threshold for the performance metric or the point at which further variable removal significantly degrades the model. The initial inclusion of all possible predictors in backward elimination ensures a thorough examination of variables, which is a key advantage of this method. By systematically removing variables, the model undergoes an iterative refinement, potentially leading to a more optimised set of features. This method's flexibility extends to its applicability across various statistical and machine learning models, enhancing its utility (Simplilearn, 2023).

A significant benefit of backward elimination is the reduced risk of overlooking important variables, a common issue in forward selection due to premature decisions on feature exclusion. However, backward elimination is not without its drawbacks. It can be computationally demanding, especially with a large number of features. There is also the risk of path dependency, where the sequence of variable removal can influence the final model, possibly leading to suboptimal results. Additionally, starting with all variables can result in overfitting, particularly when the feature count significantly exceeds the number of observations (Singh, H., 2021).

3.6. L1 Regularisation

L1 regularisation adds a penalty equal to the absolute magnitude of coefficients. It works by reducing some coefficients to zero, thus performing feature selection and keeping only essential features in the model. This is especially beneficial in high-dimensional contexts like genomic data analysis, where picking the right features is crucial. The choice of the regularisation parameter, λ, is key and usually determined through cross-validation for best results (Anuja Nagpal, 2017).

However, L1 regularisation can struggle with highly correlated variables and managing a large number of features. Its natural ability to select features leads to sparse models, which are effective in preventing overfitting in high-dimensional data and are easier to interpret. These models are useful in understanding the impact of individual features, making them an important asset in data-driven fields (Collimator, 2023).

Chapter 4: Experiment

4.1. Experiment Design

4.1.1 Purpose

The purpose of the experiments is the evaluation of the feature engineering pipeline, the assessment of the performance of different classifiers and the extraction of optimal results.

4.1.2 Introduction of Classifier

Support Vector Machine (SVM) is a robust supervised learning algorithm, effective in both classification and regression tasks. At its core, SVM seeks to identify an optimal hyperplane that distinctly separates different classes in a dataset, with the goal of maximising the margin between data points from varied classes. Its effectiveness is particularly notable in high-dimensional spaces, making it ideal for applications with numerous features. Versatile in nature, SVM can process both linear and non-linear data through the use of different kernel functions, and is recognized for its high accuracy and a lower propensity for overfitting. Nonetheless, selecting the appropriate kernel and its parameters presents a considerable challenge. SVMs are less effective with extremely large datasets, as they require lengthy training times, and they may not perform well in the presence of noisy data or when the target classes overlap significantly. Additionally, SVMs offer less interpretability compared to some other models, which can complicate the understanding of their decision-making processes (Gandhi, R., 2018).

The k-Nearest Neighbors (kNN) classifier is a simple, non-parametric approach commonly used for both classification and regression tasks. This method classifies a data point by considering the categories of its 'k' closest neighbours. The primary advantage of kNN lies in its straightforwardness and ease of use, along with its ability to adjust to varying data patterns. Its flexibility is evident as it capably addresses both classification and regression challenges without necessitating training, which can notably conserve time in certain scenarios. Nevertheless, kNN encounters difficulties with large datasets as it needs to calculate distances to all other points, posing scalability issues. Additionally, its performance diminishes with an increase in the number of dimensions, a phenomenon known as the curse of dimensionality. kNN is also prone to being influenced by noise and outliers in the dataset and demands substantial computational resources during execution for distance computations.

Forward selection is a methodical approach in feature selection that begins with a null model, which contains no predictors or features. In this process, the algorithm sequentially considers adding each feature that is not currently in the model (IMB, 2023).

Multiple Instance Learning (MIL) is an innovative form of supervised learning that centres on labelling groups of instances, known as bags, rather than individual instances. This technique proves invaluable in situations where individual instance labelling is impractical, but labelling a set of instances is achievable. In MIL, if a bag contains at least one positive instance, it is labelled positive; otherwise, it receives a negative label. A significant challenge in MIL is the ambiguity involved in identifying which specific instances within a positively labelled bag contribute to its positive status. MIL excels in handling this ambiguity of instance labelling and has broad applicability across various fields. It plays a crucial role in drug discovery, aiding in the identification of active structures in compounds where the active components are not distinctly known. In the realm of image classification and annotation, MIL helps in determining the crucial elements within an image that influence its classification. Similarly, in text categorization, MIL proves effective for classifying entire documents where only the document as a whole is labelled, rather than its individual sections (Maia P, 2023).

4.2. Dataset

4.2.1 Introduction

The data is preprocessed brain EEG data from SCI patients recorded while resting with eyes closed (EC) and eyes opened (EO). The details are as follows: the data were recorded at 250Hz through 48 electrodes and were designed to differentiate between two groups of subjects,10 people who developed pain and 8 people who did not, as to whether they would develop neuropathic pain over a 6-month period. Moreover, the data have been denoised, normalised, temporally segmented, band power estimated, and been dealt with normalised alpha, beta, theta band power while eyes closed, eyes opened, and taking the ratio of EO/EC. In addition, these data were collated into a CSV-formatted table ('data.csv') containing 180 rows (10 repetitions per subject) and 432 columns (9 features multiplied by 48 electrodes) and arranged in order of subject and primary feature type. Finally, feature identifiers were stored in 'feature_names.csv', while the category labels (0 or 1) for each row of data in 'data.csv' were recorded in 'labels.csv'.

4.2.2 Data Preprocessing

When preparing the data processing, the idea was to find a representative value which would efficiently summarise the electroencephalography (EEG) data recorded from 48 electrodes. The main purpose of our strategy was reducing the dimension of the data, and at the same time to maintain the integrity of the key information as much as possible. Therefore, we used the mean, standard deviation, maximum value, and coefficient of variation as our options.

We used preprocessing in the chi-squared test, distance correlation and L1 regularisation. For the chi-squared test, we applied the mean, standard deviation, and maximum, in addition designing a new method based on the data, which was to use the mean for the first six features and the standard deviation for the last three features. Turning to the distance correlation method, we designed one besides mean, standard deviation, and maximum, which was to use the mean for the first 6 features and the coefficient of variation for the last 3 features. Finally, for L1 regularisation, we designed a method of preprocessing using chi-square validation first.

There was not any preprocessing we used in Pearson correlation and forward-backward search methods.

Figures 1, 2 and 3 show our visualisations using Pearson's test, forward search and L1 regularisation. Normally, a discriminative feature would have a distribution that does not overlap between the categories. If a feature is predominantly concentrated within a certain value or range of values in one category and more widely distributed in another, this indicates that the feature is useful for categorisation.ECDFs can show the shape and variability of the distribution of a feature. The ideal feature is one whose ECDF curves are well separated across categories. If the ECDF curves of different categories rarely cross at a point or region, this means that the feature has good classifying potential at that point or region. Look for clear trends or separable groups of dots. It may mean that the features can help distinguish between categories if data points from different categories form distinct groups in the plot. Good combinations of features will tend to show up in scatter plots as widely spaced data points between categories.

4.3 Classifier Settings

4.3.1 SVM Parameters

In our experiment, in order to optimise the performance of the Support Vector Machine(SVM), we used a grid search method to tune parameters of the SVM.

‘svc__C’: this parameter controls the strength of the regularisation of the SVM model, and its value determines how much the data points are penalised by classification error. In our grid search, the candidate values include [0.1, 1, 10, 50, 100], which would lead to the situation that smaller values of C result in lower penalties, allowing more misclassification, while larger values of C imply stronger regularisation, reducing misclassifications but potentially leading to overfitting.

'svc__gamma': this parameter determines the range of influence of the feature space in the Radial Basis Function (RBF) kernel. We set the values to [0.001, 0.01, 0.1, 1]. Smaller values of gamma imply a wider range of feature space, resulting in smoother decision boundaries, while larger values of gamma produce more complex and closed decision boundaries.

‘svc__kernel': this parameter is used to specify the type of kernel function used by the SVM. In our experiments, we consider three kernel functions: 'rbf' (radial basis function), 'linear' (linear kernel) and 'poly' (polynomial kernel).

4.3.2 KNN Parameters

As with the SVM, we also used a grid search to tune the parameters.

'n_neighbors': this parameter determines the number of neighbours considered in the KNN algorithm. We chose these different values of [2, 3, 5, 7, 10] for testing.

'weights': this parameter is used to determine the weights of the neighbours in the calculation. We tested two weighting schemes: 'uniform' and 'distance'. In 'uniform' mode, all neighbours have the same impact, while in 'distance' mode, closer neighbours have a greater impact on the classification results, which may help to improve the performance of the model under uneven data distribution.’

'algorithm': this parameter specifies the algorithm used to compute the nearest neighbours. We considered four algorithms: 'auto', 'ball_tree', 'kd_tree' and 'brute '.

4.3.3 MIL Parameters

To optimal performance, we still applied the grid search the same as other methods.

'hidden_layer_sizes': this parameter is used to define the size and number of hidden layers in the neural network. We chose different configurations, including single layers (50 or 100 neurons) and double layers (e.g. 50-50 or 100-50 neurons).

'alpha': this parameter indicates the strength of the regularisation term, which is used to prevent overfitting. We tested different levels of alpha values including [0.0001, 0.001, 0.01, 0.1]. Smaller alpha values provide less regularisation, while larger values enhance the regularisation effect and may help to improve the model's performance on unseen data.

'activation': The activation function determines the output of the neuron. We considered three different activation functions: 'tanh', 'relu' and 'logistic'.

'solver': this parameter specifies the optimisation algorithm, which is used to train the neural network. We consider the algorithms 'sgd' (stochastic gradient descent) and 'adam'.

'batch_size': batch size determines the number of samples used in each iteration. We chose three different batch sizes [16, 8, 4] to test their impact on the learning process. Smaller batch sizes may lead to faster convergence, but may also be more unstable.

4.4 Parameters Analysis

4.4.1 Analysing the Parameters of SVM with Distance Correlation

Based on the data in Table 4, we can analyse the effect of different parameters in the Support Vector Machine (SVM) on the accuracy as follows:

In the first region, we see that ‘svc__C’ and ‘svc__kernel’ remained unchanged, while the value of ‘svc__gamma’ changed from 0.001 to 1. In this series of experiments, using the mean as the preprocessing method, with the data dimension of 180*9, the ‘svc__C’ of 0.1, and the ‘svc__kernel’ of RBF, the accuracy basically stayed around 55.56%, except when ‘svc__gamma’ is 0.1, the accuracy slightly decreased to 53.89%. This indicated that with the RBF kernel, ‘svc__kernel’ had little effect on the accuracy in this range.

In the second region, ‘svc__C’ and ‘svc__gamma’ remained constant at 0.1 and 0.001, respectively, but the kernel function changed from RBF to linear and then poly. We can observe that when the kernel was changed from RBF to linear, the accuracy improved significantly to 81.12%, and with the poly kernel the accuracy returned to 55.56%, like the RBF kernel. This change highlights the importance of kernel choice on SVM performance and suggests that the linear kernel is more effective than either the RBF or poly kernel on this dataset.

The last region shows the effect of the ‘svc__C’ parameter, here the ‘svc__kernel’ and ‘svc__gamma’ remained constant at linear and 0.001 respectively, while ‘svc__C’ varies from 0.1 to 50. In this setup, we see that the accuracy increased from 81.12% to 85.56% as the ‘svc__C’ increased and then dropped back to 81.12% again.

4.4.2 Analysing the Parameters of KNN with Distance Correlation

In Table 7, we can see how the accuracy of the model is affected by different parameters, in the KNN algorithm.

In the first region, the algorithm ('algorithm') and the number of neighbours ('n_neighbors') remained the same, while the weights ('weights') changed. Specifically, when the chi-squared test was used as a preprocessing method, the data dimension was 180*9, the 'algorithm' was chosen as ball_tree, and the 'n_neighbors' is 3, changing the 'weights' from uniform to distance results in a slight decrease in accuracy from 61.67% to 60.56%. This indicates that the choice of 'weights' has a slight effect on the accuracy in this configuration.

In the second region, the 'weights' and the 'n_neighbors' were kept constant at uniform and 3 to examine the effect of different algorithms. The three algorithms - ball_tree, kd_tree, and brute

- had unchanged accuracy of 61.67% in this particular setting. This means that the 'algorithm' choice has no effect on model performance with uniform weights and a neighbour count of 3.

The last region examined the effect of the 'n_neighbors', where the 'algorithm' and 'weights' remained the same, ball_tree and distance, respectively. The 'n_neighbors' was increased from 2 to 10 and we can see that the accuracy increased from 72.21% to 73.88% and then decreased slightly. This shows that the 'n_neighbors' has a significant effect on the accuracy of the KNN model.

4.4.3 Analysing the Parameters of MIL with Pearson Test

In Table 8, we can see how different parameters in the Multiple Instance Learning (MIL) model affect the accuracy of the model.

In the first region, when the activation function ('activation'), regularisation parameter ('alpha'), batch size ('batch_size'), and hidden layer sizes ('h_layer_sizes') were kept constant, varying the solver ('solver') from stochastic gradient descent (SGD) to adaptive moment estimation (adam) led to an increase in accuracy from 78.34% to 86.12%, which showed that the choice of solver had a significant impact on the model's performance to 86.12%. It indicates that the choice of solver has a significant impact on the model performance.

The second region shows that different choices of activation function also have different effects on the accuracy when other parameters are fixed. The accuracy using the 'tanh' function was 80.57%, while it was slightly higher than 81.12% when using the 'relu' function and was 79.46% when using the 'logistic' function. This shows that the 'relu' activation function performs better compared to the other functions on this dataset.

The third region considers the effect of 'h_layer_size'. There was no significant difference in the accuracy of one hidden layer (50 neurons) compared to the structure of two hidden layers (50 neurons and 50 neurons), both around 83.89%, while keeping other parameters constant. This indicates that increasing the number of hidden layers or the number of neurons does not have a significant effect on the accuracy in some cases.

In the fourth region, keeping the other parameters constant, by varying the regularisation parameter 'alpha', we can observe that the accuracy decreased from 83.34% to 78.89% as the alpha value increased, suggesting that stronger regularisation (higher 'alpha' values) may have a detrimental effect on the model performance.

The last region investigates the effect of 'batch_size'. With all other parameters held constant, the 'batch_size' was reduced from 16 to 4 and the accuracy improved, from 85.01% to a maximum of 83.89%. This may indicate that smaller 'batch_sizes' could help the model learn better during training.

4.5 Feature and Classifier

4.5.1 Filtering Methods

The optimal performance of the different filtering methods on SVM is shown in Table 1. The Pearson correlation achieved 91.11% accuracy in 180 * 180 dimensions, which may be due to the fact that the Pearson correlation efficiently reveals the linear relationship between the variables, which was beneficial for the linear kernel of SVM. However, it may not be applicable to capture complex nonlinear patterns, which may be a limitation when using the RBF kernel. Moreover, the distance correlation reached 93.31% accuracy in 180 * 180 dimensions when used in conjunction with SVM. This method considered the non-linear dependencies between the variables, which can capture more complex patterns when combined with the RBF kernel of SVM, improving the accuracy. However, it performed poorly in low dimensions (180 * 9), possibly because nonlinear patterns were not apparent in low dimensions, or the data was not sufficient to support the learning of a complex model. Chi-squared test had the highest accuracy of 88.88% when combined with SVM in 180 * 90 dimensions, which may be due to the fact that features associated significantly were selected with target variations, helping the SVM to perform effective classification on medium-sized datasets. However, on the low dimensional data of 180 * 9, its performance degradation may be due to the lack of sufficient features to support the model's decisions.

As shown in Table 5, which exhibits the optimal performance of the filtering methods on KNN. Pearson correlation did not perform as well in KNN as it did in SVM, with an accuracy of 74.44% in 180 * 180 dimensions, which suggested that KNN may be subject to the curse of dimensionality when dealing with high-dimensional data, resulting in a performance degradation. As for distance correlation method, when merged with KNN, it didn't perform optimally compared to its individual application. However, when integrated with distance weights, it attained a peak accuracy of 76.11% in 180 * 90 dimensions. This indicated that distance correlation's nonlinear relationship might be beneficial in KNN, but primarily in moderate dimensional settings. Finally, Chi-squared test method performed similarly in KNN, reaching 73.88% accuracy in 180 * 90 dimensions.

The optimal accuracies of all filtering methods on MIL are pooled as shown in Table 9. The Pearson correlation combined with MIL achieved 89.99% accuracy in 180 * 180 dimensions, which may be attributed to the fact that MIL is able to learn rich patterns from the linear relationship provided by the Pearson correlation. Moreover, distance correlation combined with MIL combination reached the highest accuracy of 92.22% in 180 * 180 dimensions, indicating that the nonlinear features extracted by distance correlation combined with the pattern recognition ability of MIL can achieve highly accurate classification. In addition, Chi-squared test method combined with MIL reached 91.11% accuracy in 180 * 90 dimensions, indicating that it was medium dimensionality is effective in selecting key features, which is beneficial to the MIL model.

In summary, Pearson correlation performs best in MIL, probably because MIL is suitable for learning from large-scale linear relationships; Method of distance correlation performs best in SVM, probably due to the fact that SVM is suitable for capturing complex data patterns using a nonlinear kernel. Meanwhile, Chi-squared performs well in both MIL and KNN, probably because of their ability to learn from statistically significant correlated features.

4.5.2 Wrapper Methods

As shown in Table 2, it is the best results achieved by different wrapper methods in SVM. The forward selection achieved high accuracies of 94.44% and 95.55%, respectively, when a linear kernel was used in the SVM for both the 180*90 and the 180*180 dimensions of the data. This may be due to the fact that the forward selection method kept the model simple and avoided overfitting when adding features, and the linear kernel was well suited to handle the feature set in this case. Backward selection also performed well on two different data dimensions when using the RBF kernel. However, it was more accurate on the higher dimensional data of 180*180, being 92.79%. This indicated that backward selection was able to improve the performance of the model in high dimensions by removing unimportant features, and the RBF kernel was able to make effective use of this filtered feature set.

As shown in Table 10, which demonstrates the optimal performance of the wrapper methods on MIL. Forward selection achieved 93.89% accuracy using the logistic activation function for the 180*90 data dimension and 92.77% accuracy using the 'tanh' activation function for the 180*180 data dimension. This may be owing to the fact that forward selection was able to gradually build an efficient feature set suitable for MIL to learn discriminative patterns when dealing with data of different sizes. The backward selection was slightly less accurate than the forward selection method using 'relu' and 'tanh' activation functions for data dimensions of 18090 and 180180, respectively. Backward selection may not be as effective as forward selection in MIL because it may prematurely exclude certain features that are useful for the model to learn complex relationships.

In combination, forward feature selection achieves high accuracy in both SVM and MIL, probably because it constructs the feature set in a conservative way, ensuring at each step that the features added improve the model performance. Whereas, backward feature selection performs well in SVM with the RBF kernel, probably because the RBF kernel is able to learn enough information from the reduced feature set. In MIL, backward selection may result in the removal of features that contribute to model performance improvement, which explains why forward selection is more accurate in most cases.

4.5.3 Embedding Methods

In Tables 3, 6 and 11, the L1 regularisation method combined with two preprocessing methods, mean value and Chi-square test, are applied to SVM, KNN and MIL classifiers. In the comparison, we can see that the accuracy of using the cardinality test as a preprocessing method is generally higher than that of using the mean value method.

For SVM, the combination of using L1 and Chi-square test achieved an accuracy of 95.01%, which is higher than the 87.78% using the mean method. This suggests that the Chi-square test is effective in improving the classification performance of SVM on high-dimensional data, because SVM depends on boundary decisions, and the Chi-square test clarifies these boundaries by selecting more relevant features.

For the KNN algorithm, the accuracy of the two methods is not much different, with the average method getting 72.22% and the chi-square test getting 72.77%.

For the MIL algorithm, the chi-square method improved its accuracy from 83.89% to 92.77% compared to the mean method.

In conclusion, the Chi-square preprocessing method is particularly suitable for improving the performance of the classifier when the feature dimensionality is high, due to its advantage in feature selection. On the other hand, the mean preprocessing method may be sufficient for use when the feature dimensionality is low or the classifier is not sensitive to feature selection.

Chapter 5: Conclusion

This study showcases the employing electroencephalography (EEG) data in conjunction with sophisticated feature engineering and machine learning techniques to foresee the onset of central neuropathic pain (CNP) in individuals with spinal cord injury (SCI). The investigation utilised a range of statistical and machine learning approaches, including the Chi-Square Test, Pearson Correlation, Distance Correlation, Forward Selection, Backward Elimination, and L1 Regularization, for the analysis of complex EEG data. The application of machine learning algorithms such as Support Vector Machine (SVM), k-Nearest Neighbors (kNN), and Multiple Instance Learning (MIL) notably augmented the study's efficacy.

The study yielded encouraging results, demonstrating that EEG data can be effectively utilised to identify features pertinent to the development of central neuropathic pain (CNP). By applying feature engineering techniques, the dataset was optimised, thereby enhancing the predictive capability of the model. The models proficiently differentiated between spinal cord injury (SCI) patients who developed CNP and those who did not, showcasing notable accuracy and reliability. This success marks a significant advancement in the potential for early diagnosis and intervention in CNP, offering a substantial improvement in treatment outcomes for SCI patients. The method with the highest accuracy is forward selection wrapper method in SVM.

The study highlights the crucial role of integrating diverse statistical methods and machine learning techniques for managing intricate medical data. Employing multiple methodologies facilitated a holistic approach to data analysis, leading to the creation of a more precise predictive model. Additionally, the research underscores the expansive potential of EEG data in medical data analysis, emphasising its significant applications in both neuroscience and clinical practice.

Chapter 6: Reference

Anuja Nagpal, 2017. L1 and L2 Regularization Methods. Towards Data Science. Available at: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c [Accessed 30 Nov. 2023].

Collimator, 2023. What is L1 regularization? Collimator. Available at: https://www.collimator.ai/reference-guides/what-is-l1-regularization [Accessed 30 Nov. 2023].

Das, R., Kasieczka, G., & Shih, D., 2022. Feature Selection with Distance Correlation. arXiv. Available at: https://ar5iv.org/abs/2212.00046 [Accessed 30 Nov. 2023].

Gandhi, R. (2018) Support Vector Machine - introduction to machine learning algorithms, Medium. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Accessed: 30 November 2023]).

IBM (2023) What is the K-nearest neighbors algorithm?, IBM. Available at: https://www.ibm.com/topics/knn [Accessed: 01 December 2023].

Maia, P. (2023) An introduction to multiple instance learning, NILG.AI. Available at: https://nilg.ai/202105/an-introduction-to-multiple-instance-learning/ [Accessed: 30 November 2023].

Mayo, M. (2018). Step Forward Feature Selection: A Practical Example in Python. KDnuggets. Available at: https://www.kdnuggets.com/2018/06/step-forward-feature-selection-python.html [Accessed 30 Nov. 2023].

R. V. (2022) Feature selection - correlation and P-value, Medium. Available at: https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf [Accessed: 30 November 2023].

S.M. Gajawada, 2019. Chi-Square Test for Feature Selection in Machine Learning. Towards Data Science. Available at: https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223 [Accessed 30 Nov. 2023].

Simplilearn, 2023. What Is Backward Elimination Technique In Machine Learning?. Simplilearn. Available at: https://www.simplilearn.com/what-is-backward-elimination-technique-in-machine-learning-article [Accessed 30 Nov. 2023].

Singh, H. (2021). Backward Feature Elimination and its Implementation. Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/04/backward-feature-elimination-and-its-implementation/ [Accessed 30 Nov. 2023].

Singh, H. (2021). Forward Feature Selection | Implementation of Forward Feature Selection. Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/ [Accessed 30 Nov. 2023].

sklearn, 2023. scikit-learn. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.ch [Accessed 30 Nov. 2023

1 Appendix

Table 1: Comparison of best results using filtering methods in SVM

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			SVC_C	SVC_γ	kernel	
dCor	max	$180 * 9$	50	0.1	rbf	78.34 %
dCor	mean	$180 * 9$	50	0.1	rbf	83.34 %
dCor	std	$180 * 9$	100	0.01	rbf	73.34 %
dCor	mean & CV	$180 * 9$	1	0.1	rbf	76.67 %
dCor	✘	$180 * 90$	10	0.01	rbf	88.33 %
dCor	✘	$180 * 180$	0.1	0.001	linear	93.31 %
chi-squared	max	$180 * 9$	50	0.1	rbf	78.34 %
chi-squared	mean	$180 * 9$	50	0.1	rbf	83.35 %
chi-squared	std	$180 * 9$	100	0.01	rbf	73.33 %
chi-squared	mean & std	$180 * 9$	1	0.1	rbf	73.34 %
chi-squared	✘	$180 * 90$	10	0.01	rbf	88.88 %
chi-squared	✘	$180 * 180$	100	0.001	rbf	92.23 %
pearson	✘	$180 * 90$	10	0.01	rbf	88.33 %
pearson	✘	$180 * 180$	10	0.01	rbf	91.11 %

Table 2: Comparison of results with different wrapper methods in SVM

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			SVC_C	SVC_γ	kernel	
forward	✘	$180 * 90$	50	0.01	linear	94.44 %
forward	✘	$180 * 180$	50	0.01	linear	95.55 %
backward	✘	$180 * 90$	50	0.01	rbf	88.88 %
backward	✘	$180 * 180$	50	0.01	rbf	92.79 %

Table 3: Comparison of results with different embedding methods in SVM

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			SVC_C	SVC_γ	kernel	
L1	mean	$180 * 9$	10	0.1	rbf	87.78 %
L1	chi-squared	$180 * 193$	50	0.001	rbf	95.01 %

Table 4: Comparison of results using dCor with different parameters in SVM

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			SVC_C	SVC_γ	kernel	
dCor	mean	$180 * 9$	0.1	0.001	rbf	55.56 %
dCor	mean	$180 * 9$	0.1	0.01	rbf	55.56 %
dCor	mean	$180 * 9$	0.1	0.1	rbf	53.89 %
dCor	mean	$180 * 9$	0.1	1.0	rbf	55.56 %
dCor	✘	$180 * 90$	0.1	0.001	rbf	55.56 %
dCor	✘	$180 * 90$	0.1	0.001	linear	81.12 %
dCor	✘	$180 * 90$	0.1	0.001	poly	55.56 %
dCor	✘	$180 * 90$	0.1	0.001	linear	81.12 %
dCor	✘	$180 * 90$	1.0	0.001	linear	85.56 %
dCor	✘	$180 * 90$	10.0	0.001	linear	81.12 %
dCor	✘	$180 * 90$	50	0.001	linear	81.12 %
dCor	✘	$180 * 90$	50	0.001	linear	81.12 %

Table 5: Comparison of best results using filtering methods with different parameters in KNN

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			algorithm	n_neighbors	weights	
dCor	max	$180 * 9$	auto	7	uniform	63.33 %
dCor	mean	$180 * 9$	auto	10	distance	66.11 %
dCor	std	$180 * 9$	auto	2	distance	61.67 %
dCor	mean & CV	$180 * 9$	auto	10	distance	76.11 %
dCor	✘	$180 * 90$	auto	3	distance	71.11 %
dCor	✘	$180 * 180$	auto	2	distance	76.67 %
chi-squared	max	$180 * 9$	auto	7	uniform	63.33 %
chi-squared	mean	$180 * 9$	auto	10	distance	66.11 %
chi-squared	std	$180 * 9$	auto	2	distance	61.67 %
chi-squared	mean & std	$180 * 9$	auto	2	distance	66.11 %
chi-squared	✘	$180 * 90$	auto	3	distance	73.88 %
chi-squared	✘	$180 * 180$	auto	2	distance	76.11 %
pearson	✘	$180 * 90$	auto	2	distance	70.00 %
pearson	✘	$180 * 180$	auto	3	distance	74.44 %

Table 6: Comparison of best results using embedding methods in KNN

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			algorithm	n_neighbors	weights	
L1	mean	$180 * 9$	auto	7	distance	72.22 %
L1	chi-squared	$180 * 193$	auto	10	distance	72.77 %

Table 7: Comparison of results using Chi-Squared with different parameters in KNN

Method	Preprocess Method	Data Dimension	Parameters			Accuracy
			algorithm	n_neighbors	weights	
chi-squared	mean	$180 * 9$	ball_tree	3	uniform	61.67 %
chi-squared	mean	$180 * 9$	ball_tree	3	distance	60.56 %
chi-squared	mean	$180 * 9$	ball_tree	3	uniform	61.67 %
chi-squared	mean	$180 * 9$	kd_tree	3	uniform	61.67 %
chi-squared	mean	$180 * 9$	brute	3	uniform	61.67 %
chi-squared	✘	$180 * 90$	ball_tree	2	distance	72.21 %
chi-squared	✘	$180 * 90$	ball_tree	3	distance	73.88 %
chi-squared	✘	$180 * 90$	ball_tree	5	distance	72.21 %
chi-squared	✘	$180 * 90$	ball_tree	7	distance	73.88 %
chi-squared	✘	$180 * 90$	ball_tree	10	distance	72.77 %

Table 8: Comparison of results using Pearson with different parameters in MIL

Method	Preprocess Method	Data Dimension	Parameters					Accuracy
			activation	alpha	batch_size	h_layer_sizes	solver	
pearson	✘	$180 * 90$	relu	0.0001	16	$(50,)$	sgd	78.34 %
pearson	✘	$180 * 90$	relu	0.0001	16	$(50,)$	adam	86.12 %
pearson	✘	$180 * 90$	tanh	0.001	4	$(100,)$	adam	80.57 %
pearson	✘	$180 * 90$	relu	0.001	4	$(100,)$	adam	81.12 %
pearson	✘	$180 * 90$	logistic	0.001	4	$(100,)$	adam	79.46 %
pearson	✘	$180 * 90$	relu	0.001	4	$(50,)$	adam	83.89 %
pearson	✘	$180 * 90$	relu	0.001	4	$(100,)$	adam	86.12 %
pearson	✘	$180 * 90$	relu	0.001	4	$(50, 50)$	adam	83.89 %
pearson	✘	$180 * 90$	relu	0.001	4	$(100, 50)$	adam	80.57 %
pearson	✘	$180 * 90$	relu	0.0001	4	$(50,)$	adam	83.34 %
pearson	✘	$180 * 90$	relu	0.001	4	$(50,)$	adam	83.89 %
pearson	✘	$180 * 90$	relu	0.01	4	$(50,)$	adam	79.45 %
pearson	✘	$180 * 90$	relu	0.1	4	$(50,)$	adam	78.89 %
pearson	✘	$180 * 90$	relu	0.001	16	$(50,)$	adam	85.01 %
pearson	✘	$180 * 90$	relu	0.001	8	$(50,)$	adam	82.24 %
pearson	✘	$180 * 90$	relu	0.001	4	$(50,)$	adam	83.89 %

Table 9: Comparison of results with different filtering methods in MIL

Method	Preprocess Method	Data Dimension	Parameters					Accuracy
			activation	alpha	batch_size	h_layer_sizes	solver	
dCor	max	$180 * 9$	relu	0.001	4	$(100, 50)$	adam	77.79 %
dCor	mean	$180 * 9$	relu	0.0001	8	$(50, 50)$	adam	78.34 %
dCor	std	$180 * 9$	tanh	0.0001	16	$(100,)$	adam	71.11 %
dCor	mean & CV	$180 * 9$	relu	0.001	8	$(100, 50)$	adam	78.89 %
dCor	✘	$180 * 90$	relu	0.001	4	$(50,)$	adam	83.89 %
dCor	✘	$180 * 180$	relu	0.0001	16	$(100,)$	adam	92.22 %
chi-squared	max	$180 * 9$	logistic	0.001	16	$(50,)$	adam	74.45 %
chi-squared	mean	$180 * 9$	relu	0.0001	8	$(100, 50)$	adam	78.89 %
chi-squared	std	$180 * 9$	relu	0.01	4	$(100, 50)$	sgd	70.55 %
chi-squared	mean & std	$180 * 9$	relu	0.001	8	$(100,)$	adam	82.23 %
chi-squared	✘	$180 * 90$	relu	0.0001	16	$(100,)$	adam	91.11 %
chi-squared	✘	$180 * 180$	relu	0.0001	16	$(100,)$	adam	87.22 %
pearson	✘	$180 * 90$	relu	0.0001	16	$(50,)$	adam	86.11 %
pearson	✘	$180 * 180$	relu	0.001	4	$(50, 50)$	adam	89.99 %

Table 10: Comparison of results with different wrapper methods in MIL

Method	Preprocess Method	Data Dimension	Parameters					Accuracy
			activation	alpha	batch_size	h_layer_sizes	solver	
forward	✘	$180 * 90$	logistic	0.0001	4	$(50, 50)$	adam	93.89 %
forward	✘	$180 * 180$	tanh	0.0001	8	$(50,)$	adam	92.77 %
backward	✘	$180 * 90$	relu	0.0001	4	$(50,)$	adam	83.34 %
backward	✘	$180 * 180$	tanh	0.0001	4	$(100,)$	adam	87.22 %

Table 11: Comparison of results with different embedding methods in MIL

Method	Preprocess Method	Data Dimension	Parameters					Accuracy
			activation	alpha	batch_size	h_layer_sizes	solver	
L1	mean	$180 * 9$	relu	0.01	4	$(100,)$	adam	83.89 %
L1	chi-squared	$180 * 193$	tanh	0.0001	16	$(50, 50)$	adam	92.77 %

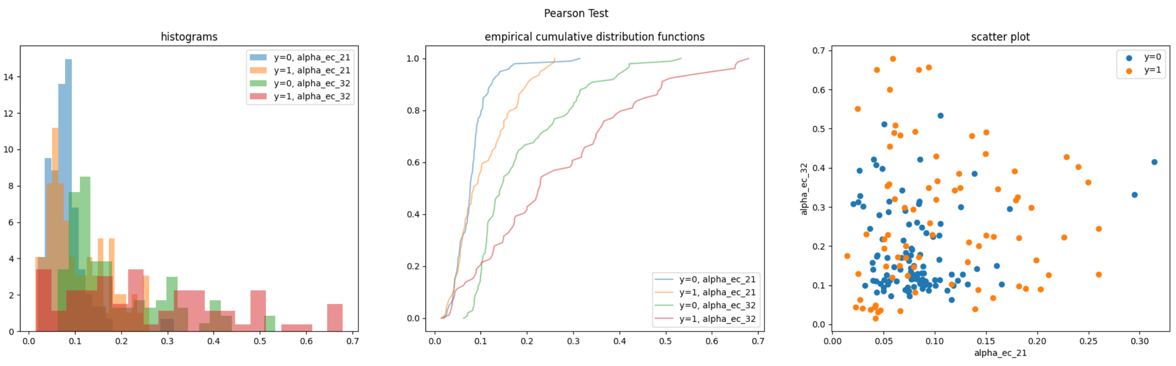

Figure 1: Pearson Test

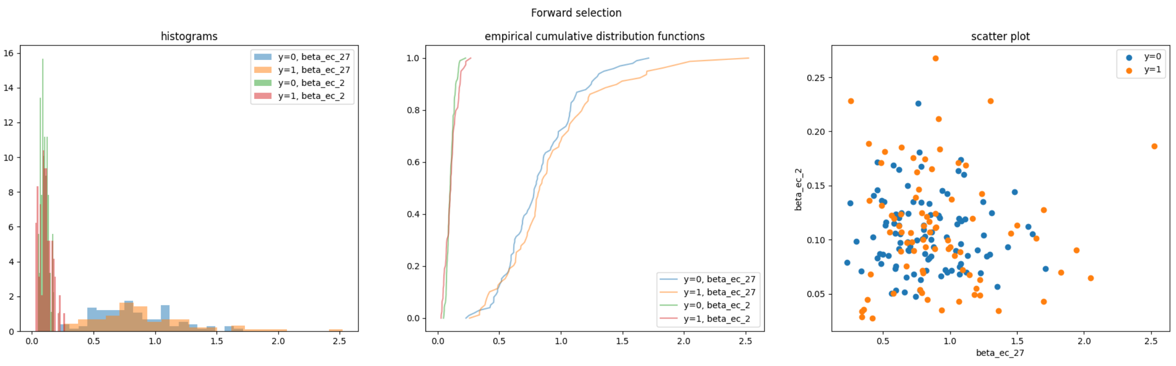

Figure 2: Forward Selection

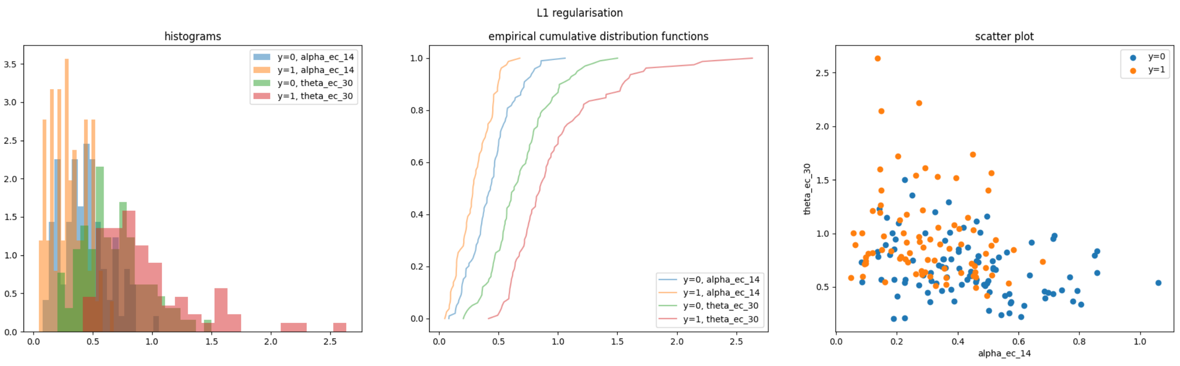

Figure 3: L1 Regularisation