M 2024

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

REVISITING CHEMOMETRICS:

FEATURE SELECTION AND CONVOLUTIONAL NEURAL NETWORKS

LUCAS ZULCHNER RIOS

MASTER THESIS IN CHEMICAL ENGINEERING
PRESENTED TO THE FACULTY OF ENGINEERING
OF THE UNIVERSITY OF PORTO

Master in Chemical Engineering

Revisiting Chemometrics: Feature Selection and Convolutional Neural Networks

Master Thesis

of

Lucas Zulchner Rios

Developed within the course of dissertation

held in

DATAHOW

Supervisor at FEUP: **Prof. Fernando Gomes Martins**

Coordinator at DataHow: **Eng. Hugo Santos**

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DEPARTAMENTO DE
ENGENHARIA QUÍMICA

July 2024

Acknowledgment

There are many people who contributed to the completion of my Master's Degree and this thesis.

Firstly, I would like to thank my supervisors, Fernando Gomes Martins and Hugo Santos, for their guidance and availability in providing knowledge and insights, which taught me much and greatly improved the final work.

To my family, who unconditionally supported me throughout this journey.

To my friends who accompanied me through these years, providing an audience for my many jokes and joyful moments.

To my dear girlfriend, Marília Montenegro, for her company, partnership, and for putting up with me for so long and listening to my rambles in times of need.

To the Debating Society of the University of Porto for all the tournaments, laughs, challenges, and moments that shaped me into a more capable person.

Lastly, I would like to thank DataHow for providing the computer on which this work was conducted.

Professor Fernando Gomes Martins, supervisor of this dissertation, is an integrated member of LEPABE – Laboratory for Process Engineering, Environment, Biotechnology, and Energy, supported by national funds through FCT/MCTES (PIDDAC): LEPABE, UIDB/00511/2020 (DOI: 10.54499/UIDB/00511/2020) and UIDP/00511/2020 (DOI: 10.54499/UIDP/00511/2020) and ALiCE, LA/P/0045/2020 (DOI: 10.54499/LA/P/0045/2020).

Abstract

This work explores the intersection of chemometrics and certain machine learning techniques that are yet uncommon in chemometrics, specifically focusing on feature selection and the application of Convolutional Neural Networks to spectral data. The study is situated within the context of the chemical and pharmaceutical industries' need for efficient and accurate real-time process monitoring and control. The primary objective is to evaluate the effectiveness of several feature selection techniques in reducing the dimensionality of spectral data and assess the improvement in predictive performance offered by Convolutional Neural Networks.

The research involves a comprehensive benchmarking of feature selection algorithms, including filter methods like Minimum Redundancy Maximum Relevance and Mutual Information, as well as wrapper methods such as Permutation Importance and Sequential Feature Selection. These algorithms were tested using three widely adopted regression models: Partial Least Squares, Artificial Neural Networks and Support Vector Machine Regression. Additionally, Convolutional Neural Networks were implemented and fine-tuned to assess their viability as an alternative modeling technique for spectral data. The study was supported in two distinct datasets, and all relevant code has been made publicly available for reproducibility and further research.

The results indicate that feature selection significantly reduces the number of features required, enhancing model simplicity and robustness. Among the evaluated methods, Mutual Information demonstrated the best balance between ease of implementation and performance, while Permutation Importance showed potential for providing the most robustness. The findings suggest that multiple optimal feature subsets exist due to the inherent redundancy in the data. Convolutional Neural Networks showed promise but require further research to optimize their application in this context. This work contributes to the advancement of chemometric techniques and supports industrial practices by proposing advancements in resources usage and data analysis efficiency.

Keywords: Chemometrics, Feature Selection, Convolutional Neural Networks, Spectral Data, Machine Learning

Resumo

Este trabalho explora a interseção entre a quimiometria e certas técnicas de aprendizagem automática que ainda são incomuns nessa área, com foco específico na seleção de características e na aplicação de Redes Neuronais Convolucionais a dados espectrais. O estudo é motivado pela necessidade das indústrias química e farmacêutica de monitorar e controlar processos em tempo real de forma eficiente e precisa. O principal objetivo é avaliar a eficácia de várias técnicas de seleção de características na redução da dimensionalidade dos dados espectrais e avaliar se uma melhora de desempenho é possível através das Redes Neuronais Convolucionais.

A pesquisa realiza uma comparação abrangente entre diferentes algoritmos de seleção de características, incluindo métodos de filtragem como Mínima Redundância Máxima Relevância e Informação Mútua, além de métodos envoltórios, como Importância por Permutação. Esses algoritmos foram testados utilizando três modelos de regressão amplamente adotados: Mínimos Quadrados Parciais, Redes Neuronais Artificiais e Regressão por Máquinas de Vetor de Suporte. As Redes Neuronais Convolucionais foram implementadas para avaliar a sua viabilidade como uma técnica alternativa para modelação de dados espectrais. O estudo utilizou dois conjuntos de dados distintos, e todo o código está disponível publicamente para garantir reprodutibilidade.

Os resultados indicam que a seleção de características pode reduzir significativamente o número de características (variáveis) necessárias, contribuindo para um modelo mais simples e robusto. Entre os métodos avaliados, a técnica de Informação Mútua demonstrou o melhor equilíbrio entre facilidade de implementação e desempenho, enquanto o método de Importância por Permutação mostrou um maior potencial para fornecer robustez. Os resultados sugerem que múltiplos subconjuntos de características atingem um ponto ótimo de poder preditivo devido à redundância inerente nos dados. As Redes Neuronais Convolucionais são promissoras, mas necessitam de mais avanços para facilitar a sua aplicação neste contexto. Este trabalho contribui para o avanço das técnicas de quimiometria e para o desenvolvimento de práticas industriais mais eficientes.

Declaration

I hereby declare, under word of honor, that this work is original and that all non-original contributions are indicated and properly referenced to the respective authors and sources.

July 2024

Contents

List of Figures

List of Tables

Glossary

Adam Adaptive Moment Estimation

ANN Artificial Neural Network

CARS Competitive Adaptive Reweighted Sampling

CNN Convolutional Neural Network

CV Cross Validation

FTIR Fourier Transform Infrared

GA Genetic Algorithms

KNN K-Nearest Neighbors

MI Mutual Information

MLR Multivariate Linear Regression

mRMR Minimum Redundancy Maximum Relevance

MSE Mean Squared Error

NIR Near-Infrared

OPS Ordered Prediction Selection

PCA Principal Component Analysis

PI Permutation Importance

PLS Partial Least Squares

PLSDA Partial Least Squares Discriminant Analysis

ReLU Rectified Linear Unit

SDGs Sustainable Development Goals

SG Savitzky–Golay

SFS Sequential Feature Selection

SVR Support Vector Machine Regression

1 Introduction

A daily challenge faced by the chemical and pharmaceutical industries is the constant measurement and control of process variables to ensure product quality and safety. This challenge involves managing the costs and reliability of real-time execution, as it often requires transporting samples to specialized equipment for accurate measurements. Chemometrics addresses this challenge by utilizing a data-driven approach to extract properties from chemical systems, providing a cost-effective and reliable alternative to the traditional laborious approach.

Chemometrics employs various spectroscopic techniques, which use the interaction of matter with electromagnetic radiation to obtain information about the chemical composition and properties of a sample. Techniques such as near-infrared (NIR), Fourier transform infrared (FTIR), and Raman spectroscopy are commonly used as indirect measures of these properties.

As spectroscopic techniques have become increasingly integral to industrial practices, the importance of effectively managing the underlying data has become evident. In the field of data science, models are developed by learning from data examples, with performance largely influenced by the quality of the data and the chosen model. Feature selection, a dimensionality reduction technique that identifies the most pertinent information for model development, is a promising approach to improve model performance, robustness, and runtime. By focusing on the specific types of data (features) that are most relevant to the problem at hand, feature selection can minimize computational resources, and eliminate redundant and noisy data prior to model training. This results in a more informative dataset, which not only helps to develop models with higher predictive capabilities but also contributes to the creation of more robust models that are less sensitive to variations in input data. By applying feature selection techniques before training the model, data scientists can optimize the use of available data, leading to improved model performance and more efficient use of computational resources.

In the field of spectral data analysis, whether for the prediction of categorical (classification tasks) or continuous (regression tasks) variables, there is consensus about redundancy in spectral data [1]. As a result, dimensionality reduction methods are frequently employed to reduce computational time, increase interpretability and most notably contribute to simpler models, as using the full spectra will result in a higher-dimensional model. These techniques are broadly categorized into two main groups [2]:

- **Feature Extraction**: This process utilizes methods such as Principal Component Analysis (PCA) and Partial Least Squares (PLS) to transform the original features into a new set of variables that capture the essential information while reducing the dimensionality of the data. These transformed features, often called components and latent variables, respectively, are linear combinations of the original features and aim to retain the most relevant characteristics of the data in a more compact representation;
- **Feature Selection**: This involves techniques that identify the most consistent, non-redundant, and relevant subset of features for model building. These techniques can be further categorized into three groups:

- **Filter methods**: These methods select features based on statistical scores that often indicate relevance to the target variable. For instance, a simple filter method is feature selection by variance, in which features with higher variance are considered more informative and, therefore, more important;
- **Wrapper methods**: These methods wrap around a predictive model, using its predictions to score features and select the best-performing subset;
- **Embedded methods**: These strategies integrate feature selection into the model training step. Examples include tree-based models such as decision trees and random forests, which assess feature importance based on the decrease in impurity (Gini). Regularization can also be considered a type of embedded feature selection strategy.

Another approach to handling spectral data is the use of a Convolutional Neural Network (CNN). In contrast to traditional linear models, CNN are more complex and can effectively capture both linear and nonlinear relationships within the data. The key operation applied is the convolution, which allows the network to learn local patterns and hierarchical features directly from the input data. By learning these patterns automatically, CNN can potentially reduce the need for preprocessing, leading to models that are more robust and easier to apply across diverse datasets without the need for extensive human tailoring [3].

The ability of CNN to learn from raw data has been fundamental to their success in tasks such as image classification in computer vision. Their architecture, designed to detect patterns and features in images, has revolutionized fields like healthcare (e.g., by detecting tumors in medical scans), autonomous driving (e.g., by recognizing pedestrians and traffic signs), and social media (e.g., by tagging friends in photos). CNN consistently achieve high accuracy and performance, making them the standard solution for image classification challenges. Their successful application to single and multidimensional data demonstrates their versatility and capability to efficiently handle different data structures.

The final goal of this study is to evaluate the extent to which dimensionality reduction can be applied to spectral data. Specifically, it seeks to determine how significantly dimensions can be reduced, assess the impact of this reduction on performance, and better understand the conditions under which dimensionality reduction is effective or ineffective. Additionally, it seeks to benchmark CNN and assess their viability as an alternative modeling technique.

1.1 Contribution of the Author to the Work

This thesis employed a benchmarking of four feature selection techniques: Sequential Feature Selection (SFS), Permutation Importance (PI), Minimum Redundancy Maximum Relevance (mRMR), and Mutual Information (MI) through three widely adopted models: Artificial Neural Networks (ANN), PLS, and Support Vector Machine Regression (SVR). These models were tested across two distinct datasets. The features selected by each algorithm were studied and compared to better understand trends. Additionally, CNN were benchmarked, fine-tuned, and compared to ANN. All relevant code has been extensively commented and is publicly available in the GitHub repository: https://github.com/Lucas-ZR/datahow-thesis-feup.

1.2 Document Structure

This chapter provides an introduction to chemometrics, followed by an exploration of feature selection and CNN as methods to enhance process variable measurement and data analysis within the chemical and pharmaceutical industries. The subsequent chapters are organized as follows:

- Chapter 2 (p. 4), **State of the Art**, describes the state of the art regarding this project's scope, identifying current trends in literature;
- Chapter 3 (p. 10), **Methods**, outlines the inner-workings behind the chosen models and feature selection algorithms while, regarding considerations and challenges of implementation;
- Chapter 4 (p. 26), **Results and Discussion**, presents and discusses the main results drawn from the benchmarking of the various feature selection techniques and the CNN;
- Chapter 5 (p. 43), **Conclusions**, concludes the dissertation with a reflection on what can be expected to be achieved with feature selection and CNN while also including a brief discussion on future work.

2 State of the Art

A literature review was conducted to gather information on the state of the art of feature selection and convolutional networks applied to spectral data. Notably, feature selection methods appear more frequently than feature extraction in existing literature. Consequently, this study prioritizes feature selection but also incorporates an examination of PLS, a technique that can be employed for feature extraction and is prevalent in the literature. To guarantee high-quality evidence, specific exclusion criteria were employed.

2.1 Feature Selection

This search was conducted during the period of realization of this dissertation. Recent and frequently cited studies were prioritized, as they were considered more relevant. The query, designed to capture the most relevant keywords without being either overly inclusive or overly exclusive, was as follows.

(“Raman spectroscopy” OR “NIR spectroscopy”) AND (“feature selection” OR “wavenumber selection”) AND regression

To accommodate variations in terminology and capture a wider range of relevant literature, slightly different versions of this query were utilized. All the queries were subjected to the following criteria:

Table 2.1: Exclusion criteria.

ID	Criterion
Exclusion	
EC1	Not written in English.
EC2	Focused on other aspects of spectroscopy, like data collection.
EC3	Works focused on image processing (considered in the next section).
EC4	Works employing feature selection algorithms that work exclusively on classification tasks.
EC5	Works employing spectroscopy techniques not commonly used in chemical and biopharmaceutical industries.

After filtering, 14 references were selected. Each employs one or more feature selection algorithms as a means to enhance predictive models in various domains, all focusing specifically on spectral processing. The works are summarized as follows.

- **Balabin et al.(2011)** [4] investigated a range of feature selection techniques on biodiesel NIR data, aimed at predicting viscosity, water content, and other properties. The study employed one filter method and several wrapper methods. Feature selection improved performance significantly on the Multivariate Linear Regression (MLR) model, while only minor improvements were seen in PLS. Additionally, the research indicated that no combination of feature selection and linear models outperformed nonlinear models such as ANN. Lastly, a significant amount of features could be removed from the original dataset;

- **Teófilo et al. (2009)** [5] proposed a novel embedded method, Ordered Prediction Selection (OPS), tailored to a PLS model. This method was applied across various spectroscopy techniques, including Raman and NIR, to predict physical proprieties such as freezing temperature and total aromatic mass of fuels. The study concluded that OPS moderately improved predictive performance while significantly reduced the total number of features included in the predictive model;

- **Xu et al. (2017)** [6]: employed several feature selection methods to predict rice root density via NIR spectroscopy with Support Vector Regression (SVR). The methods included Competitive Adaptive Reweighted Sampling (CARS, a wrapper method), Genetic Algorithms (wrapper method), Successive Projection Algorithm (filter method), and Uninformative Variable Elimination (filter method). Among these, CARS were notably effective, significantly enhancing performance while also reducing the total number of features;

- **Chen et al. (2018)** [7] utilized mRMR, a filter method, through PLS-DA (DA stands for discriminant analysis, a variation of PLS tailored for classification tasks) on milk powder NIR spectra. Observations pointed towards a significant reduction in spectra size, with no noticeable increase or decrease in performance;

- **Knief et al. (2009)** [8] employed genetic algorithms (GA) on PLS to investigate alveolar toxicity via Raman spectra. This study concluded that GA is able to achieve a slight increase in performance while moderately reduces the total number of features;

- **Zhao et al. (2016)** [9] utilized a regularization approach, known as the Least Absolute Shrinkage and Selection Operator (LASSO), which incorporates L1 norm regularization. This technique is categorized as an embedded method. Additionally, they employed stepwise regression, a type of wrapper method, for the classification of Raman skin cancer cells. While these methods offer slight improvements in classification accuracy, the extent to which features can be removed was not explicitly stated;

- **Xie et al. (2015)** [10] compared full spectra PLS and Support Vector Machines (SVM) with feature selection by CARS on FT-NIR spectra for prediction of arginine content. Although the overall improvement in predictive performance was minimal, models achieved comparable accuracy using only 2.04 % of the full spectral data;

- **Schwanninger et al. (2011)** [11] implemented a knowledge-based feature selection method, selecting specific wavenumber ranges known to be relevant for the component under analysis, using FT-NIR to estimate lignin content in wood. This approach significantly reduced the number of features in the model while maintaining its predictive capability through PLS;

- **Xu and Lu et al. (2018)** [12] implemented various feature selection techniques on Support Vector Machines (SVM) and PLS, notably Recursive Feature Elimination (RFE, wrapper) and a variant of the Mutual Information method (filter). The study focused on predicting viscosity and boiling point of fuels through NIR spectra, the methods employed notably reduced

the total number of variables used by the model with only a slight impact on predictive performance;

- **Kaneko et al. (2021)** [13] introduced the use of the Boruta algorithim, a wrapper method based on Random Forests, and Variable Importance in Projection (VIP) to enhance PLS predictions of lactate and glucose concentrations from NIR spectra. Additionally, the research incorporated transfer learning to boost predictive accuracy. While the feature selection algorithms significantly reduced the number of features, using Boruta alone negatively impacted performance. However, combining Boruta with VIP resulted in a slight improvement, notably, transfer learning enhanced performance more effectively than the feature selection algorithms alone;

- **Ma et al. (2024)** [14] evaluated the mRMR method applied to various modeling techniques — Support Vector Machines (SVM), Artificial Neural Networks, and Random Forests — using near-infrared (NIR) spectral data. The findings indicate that mRMR enables significant reduction of spectral data with minimal impact on the predictive accuracy of the models;

- **Cai et al. (2019)** [15] applied a modified version of the CARS algorithm, Stability Competitive Adaptive Reweighted Sampling (SCARS), alongside genetic algorithms and other feature selection techniques on the following models: PLS, ANN, and Support Vector Regression (SVR). These feature selection methods slightly improved predictive accuracy while significantly reduced the number of features required to predict ore content from Raman spectra;

- **Lanza et al. (2021)** [16] employed near-infrared (NIR) spectra to classify the spoilage of chicken breast meat using PLS-DA alongside the Boruta and Stepwise feature selection algorithms. The study concludes that feature selection significantly reduced the number of variables while slightly enhanced predictive performance;

- **Shizuang et al. (2014)** [17] utilized Surface-Enhanced Raman Spectroscopy (SERS), a more sensitive variation of Raman spectroscopy, which is particularly effective for analyzing solutions with low concentrations due to its increased sensitivity [18]. This spectroscopy technique was applied to predict the content of thiram through PLS. Conventional and adaptive genetic algorithms were implemented to optimize the model, which resulted in a slight improvement, the study did not specify the exact number of features reduced by the algorithms.

All references mentioned above were evaluated based on two metrics: the degree to which predictive performance improves and the degree to which the number of features is reduced. Performance improvement was categorized as minor (no improvement or less than 5 % improvement), moderate (up to 30 % increase), or significant (more than 30 % increase). Similarly, feature reduction was classified as minor (less than 20 % reduction), moderate (up to 60 % reduction), or significant (more than 60 % reduction). These metrics helped to better organize the findings in the literature and can be visualized in the following table.

Table 2.2: Summarized findings on literature review.

Predictive Improvement	Feature Reduction	References
Minor	Moderate	[8]
Minor	Significant	[4],[5],[7],[10],[11],[12],[13],[14],[15],[16]
Significant	Moderate	[4]
Significant	Significant	[6]

References [9] and [17] were excluded from the table due to the non specification of the feature reduction extent metric.

While this review is not exhaustive, it highlights several noticeable patterns:

- The overwhelming majority of the works employed PLS and achieved good predictive capability for each specific scenario, this preference is likely due to PLS's straightforward application, as it only requires the tuning of a single parameter: the number of latent variables. Additionally, its ability to filter noise and redundancy heavily contributes to its versatility, making it an almost trivial choice in predictive modeling;
- Most of the works employ filter or wrapper methods, while embedded methods are rarely seen in comparison. This observed trend might be due to the limitations intrinsic to the embedded methods, as they are often restricted to specific models such as random forests;
- Despite the clear avoidance of wrapper methods, the literature shows a relatively even exploration of alternative feature selection techniques. Apart from the considerable employment of genetic algorithms through many works, there are no clear second or third-place contenders;
- The majority of the works consistently identify a smaller subset of features that are able to perform comparably to the full set. Performance only declined notably when the subset was significantly smaller than the full spectrum, typically by at least 90 %. In a few cases, feature selection showed significant improvements in performance, particularly when applied to multivariate linear regression (MLR). This improvement could likely be attributed to the MLR model's difficulty in handling redundant and noisy data. Additionally, improvements were observed in the specific application of OPS PLS. While feature selection doesn't often boost model performance directly, it consistently reduces the total number of required features. This reduction not only doesn't compromise performance but also contributes to model simplicity, robustness, and runtime.

2.2 Convolutional Neural Networks

CNN represent a specific modeling approach, distinct from the broader field of feature selection that encompasses a variety of methods. Despite this focus, developing CNN involves making a considerable number of decisions: the choice in architecture and in hyperparameters, both of

which contain a vast range of possibilities. Often, designers rely on well-established architectures that have proven successful in practice. These architectures adhere to specific sequences of convolutional and pooling layers, optimizing the network's ability to learn and generalize from complex datasets. Popular examples include ResNets and GoogLeNets [3].

Yang et al. (2019) [3] conducted a comprehensive review of CNN applications in spectral data processing, addressing many critical decisions that affect the success of CNN development. Based on Yang's review, and variations of the following query, references were selected and filtered for relevance, prioritizing highly cited sources. The exclusion criteria are listed in Table 2.1, now with the exclusion of EC3.

("Deep Learning" OR "Convolutional Networks") AND ("spectral analysis" OR "spectroscopy")

The works are summarized as follows:

- **Acquarelli et al. (2017)** [19] performed a comprehensive analysis of classification tasks across a variety of spectroscopy techniques, including FTIR, NIR, and Raman. They compare the effectiveness of CNN against other methods like Logistic Regression, PLS-DA, and K-Nearest Neighbors (KNN), analyzing multiple datasets that vary in sample and feature sizes. Their findings reveal that CNN consistently outperform the alternatives, whether the data is preprocessed or not. This consistency highlights the adaptability and strength of CNN in handling complex spectral data;

- **Liu et al. (2017)** [20] compared various CNN architectures with commonly used models such as Random Forests, K-Nearest Neighbors (KNN), and Support Vector Machines on the RRUFF mineral dataset for classification. RRUFF is a comprehensive open-access database of spectroscopic and crystallographic data for minerals, widely used in geology and materials science. The study concluded that CNN models consistently delivered solid results, achieving higher accuracy than all other tested models, regardless of whether the data was preprocessed or used in its raw form;

- **Zhang et al. (2019)** [21] proposed the use of Inception structures—specialized convolutional layers—to predict properties such as the protein content of wheat and corn, comparing these to other CNN architectures. The novel Inception structure, named "DeepSpectra", demonstrated a slight improvement over alternative architectures. Notably, architectures that heavily relied on pooling layers performed significantly worse than others. This suggests that excessive use of pooling layers may lead to information loss, which in turn results in poorer performance;

- **Cui et al. (2018)** [22] compared a single CNN architecture, varying activation function and other parameters, with the traditional PLS approach. By analyzing the regression coefficients of the model along with performance metrics, the study concludes that CNN structures are able to consistently achieve better performance and significantly lower noise levels that the traditional approach, even in smaller datasets, indicating the robustness of CNN's in spectral modeling.

The employment of CNN on spectral data appears to be more widely regarded than the application of feature selection. The literature largely agrees that CNN, when properly configured, have the potential to achieve better results than conventional methods such as PLS. A key advantage is that the filters can also serve as preprocessing methods, allowing the network to learn the appropriate processing autonomously. This capability can significantly reduce the efforts required during the initial stages of processing [22].

In contrast to feature selection, where no specific combination of model and feature selection consistently improves accuracy, CNN emerge as potentially more precise alternatives to conventional methods such as PLS, even eliminating the need for preprocessing. Despite the complexities involved in developing CNN, there is compelling evidence that they are potentially more effective for spectral data analysis than the standard feature selection and modeling techniques. The main caveat with CNN, identified by Yang [3], is that interpretability and repeatability might be a challenge, specially on smaller datasets.

3 Methods

3.1 Datasets and Data Splitting

Two NIR datasets were selected for this study. The main characteristic of NIR spectra is their adherence to Lambert-Beer's law, which implies that the relationships between absorption and target variables are linear [13]. Consequently, linear models may be more suitable for NIR spectra than for other spectroscopic techniques.

While the most suitable models may vary for different types of spectra, the impact of feature selection on modeling is similar across spectroscopic techniques. This similarity arises because neighboring wavenumbers are often highly correlated in many spectroscopic methods [9]. Consequently, evaluating feature selection on NIR can represent its behavior in other spectroscopic methods.

Both selected datasets are relatively small, which often poses challenges due to lack of representativity. Small datasets tend to show higher variance and a higher tendency to overfit. In contrast, large datasets, while less prone to these issues, are considerably more reliant on computational power [23]. Thus, the assessment of feature selection in these small datasets is also representative of feature selection on larger spectral datasets, with the main difference being the challenges encountered during the modeling phase.

The datasets used in this study are referred to as Dataset 1 and Dataset 2. Both datasets are available on GitHub. Dataset 1 consists of near-infrared (NIR) spectra from fuel samples, with the target variable being the viscosity of the samples. It contains 401 features and 395 observations. The origin of Dataset 1 is unknown. Dataset 2 is related to maize germination rate, also evaluated through NIR spectra, and it contains 1846 features and 316 samples. Dataset 2 comes from Ma et al. (2024) [14].

A relevant aspect in chemometrics is related to establishing metrics to predict the performance of models. One crucial technique for this purpose is cross-validation. The cross-validation technique was employed in this study. Cross-validation is a commonly applied method, especially on smaller datasets, for testing the performance of predictive models. It works by subdividing the training data into different folds, and then selecting a single fold for evaluating the performance while the remaining folds are used to generate the model. This process is repeated enough times so that every fold gets used in evaluation, resulting in k distinct scores (when using k folds) that are representative of the generalization capability of the model. The scores between folds are averaged, and this metric, often denoted as CV, is used in the modeling phase (see Figure 3.1 taken from [24]).

The advantage of using cross-validation lies in its ability to provide additional evaluation metrics, such as Mean Squared Error (MSE) and the determination coefficient R^2, which offer a comprehensive assessment of the model's performance across different aspects. This method is more robust as it averages performance over multiple splits, reducing the impact of any particular split and providing a more reliable measure of the model's generalizability. Furthermore, cross-validation makes efficient use of the available data by utilizing the entire dataset for both training and validation. This ensures that the test set is reserved solely for final evaluation, while the

folds and cross-validation procedure are employed to tune and optimize the model, enhancing its robustness and reliability.

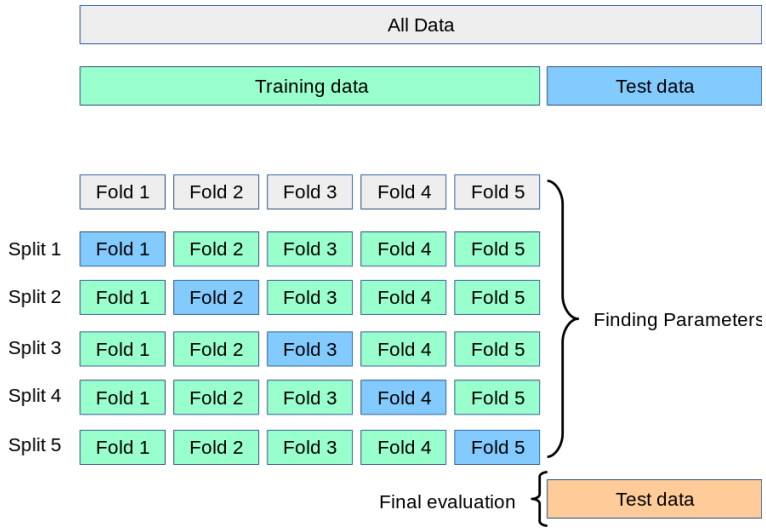


Figure 3.1: Visual representation of cross-validation, taken from Pedregosa et al. (2011) [24].

3.2 Feature Selection

Feature selection is a step in the modeling phase of machine learning and statistical modeling. It involves identifying and selecting a subset of relevant features (variables, predictors) from the data that contribute most to the predictive performance of the model. The primary goals of feature selection are to improve the model's performance, reduce overfitting, enhance generalization capability, and reduce computational complexity.

Feature selection methods can be broadly categorized into filter methods, wrapper methods, and embedded methods. The latter category was not implemented in this study because as it is not widely used.

3.2.1 Filter Methods

Filter methods are feature selection algorithms that estimate the importance of features based solely on the input X and the output Y, without employing a specific model. These methods can be utilized in various ways, with numerous algorithms available. The core concept is that features are scored based on certain criteria, and the most important features, that score better, are presumed to contribute more significantly to predictive capability.

These algorithms assign a numerical score to each feature, which is then used to select a subset of the most important features. In earlier iterations of this work, minimum-maximum scaling was applied to features' scores, and subset selection was performed through thresholds. A more elegant and suitable alternative is the use of relative ranking. Features are ranked from first to last based on their scores, and selection is made by rank. This ranking is very useful for selecting subsets because it allows the use of sorting methods, which are convenient for selecting a specified number or percentage of ranked features. This specific implementation is also useful

for comparing different feature selection algorithms and models, as the ranking is relative and easily comparable.

3.2.1.1 Minimum Redundancy Maximum Relevance

mRMR is commonly used in feature selection studies [14, 7]. It was first proposed by Ding et al. (2005) [25], addressing the problem of selecting a subset of features that are most discriminative to the target variable while also being minimally redundant with each other among the selected features.

The relevance of a feature is quantified by its mutual information with the target variable, whereas redundancy between two features is quantified by their mutual information with each other. Ding et al. (2005) [25] present a mathematical expression suitable for estimating said mutual information on discrete data points. For continuous data points, mutual information is also a valid choice, but some adjustments are required.

A comprehensive paper by Zhao et al. (2019) [26] compares different implementations of mRMR, specifically distinct ways to estimate the relevance and redundancy of features, and evaluates their performance on different datasets. The paper concludes that different implementations result in only mildly distinct outcomes.

One simple alternative used in the aforementioned paper, and employed in this work, is the assessment of relevance through an F-test, which is essentially a ratio of variances, and Pearson's correlation to assess redundancy. The scoring of each feature is then calculated by the difference or quotient between relevance and redundancy, similar to the minimization of redundancy and maximization of relevance from the original paper. The quotient is used because it produced slightly better results in the study conducted by Zhao et al. (2019) [26]. This scoring can be depicted by the following expression:

$$f(X_i) = \frac{F(Y, X_i)}{\frac{1}{|S|}\sum_{X_s \in S} \rho(X_s, X_i)} \tag{1}$$

- $f(X_i)$ represents the score of the feature X_i;
- $F(Y, X_i)$ represents the F-test between target Y and feature X_i;
- $\sum_{X_s \in S} \rho(X_s, X_i)$ represents the sum of Pearson's correlation coefficients between the feature X_i and all other features X_s in the set S, where $s \neq i$;
- $|S|$ represents the total number of other features X_s in the set S, excluding X_i itself.

The main caveat of this method is that the F-test and, especially, Pearson's coefficient are not well-suited for capturing non-linear relationships, so they may not yield consistent results when applied to highly non-linear datasets.

3.2.1.2 Mutual Information

MI is utilized here as an alternative, potentially more accurate, method for estimating the relationship between two variables compared to the F-test employed in mRMR. It specifically quantifies the dependency between the variables, being equal to zero if and only if the variables are independent, while higher values indicate higher dependency [27].

Its implementation is considerably more nuanced than the simple F-test, as it relies on non-parametric methods based on entropy estimation from k-nearest neighbors distances, as described by Kraskov et al. (2004) [28] and Ross et al. [29], both based on the idea originally proposed by Kozachenko et al. (1987) [30] [24].

Figure 3.2, taken from [24], illustrates the differences between the F-test statistic and MI. Three illustrative features x_1, x_2 and x_3 are distributed uniformly over [0,1], the target depends on them by: $y = x_1 + sin(6\pi x_2) + 0.1N(0,1)$, that is, y is completely independent of x_3. The following F-test and MI scores are normalized.

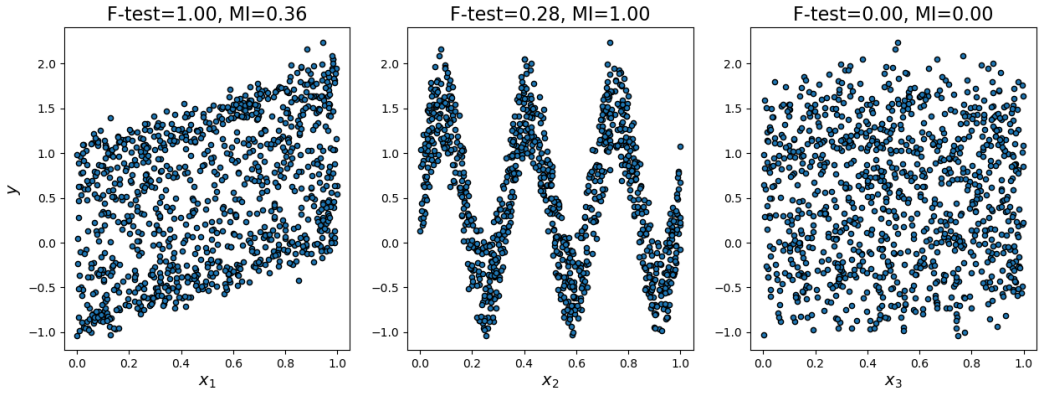


Figure 3.2: Comparison of F-test and MI, taken from Pedregosa et al. (2011) [24].

As the F-test captures only linear dependency, it rates x_1 as the most discriminative feature. On the other hand, MI can capture any kind of dependency between variables, and it rates x_2 as the most discriminative feature, which probably agrees better with human intuitive perception for this example. Both methods correctly mark x_3 as irrelevant [24].

The core difference between the application of mRMR and MI is that MI is suitable for capturing any kind of relationship between variables, linear or not, and it scores features based only on their relationship with the target (relevance) while mRMR also includes redundancy for scoring.

3.2.2 Wrapper Methods

Unlike filter methods, wrapper methods use a specified model for scoring features. Wrappers are flexible in that they work by specifically identifying the features that better contribute to the specified model's predictive performance. This means that features might be scored differently depending on the model used, which can be advantageous because different models perceive different patterns or relationships within the dataset.

The nuances of modeling referred to in the next section are of utmost importance when it comes to wrapper methods because improper modeling is likely to affect feature scoring. Overfitting or underfitting might inaccurately score features because the model is incapable of correctly using the data. Therefore, for the correct assessment of wrapper methods, "good" models, are recommended.

3.2.2.1 Permutation Importance

PI is a wrapper method that is similar to the Boruta algorithm used in recent studies [13, 16]. The core idea is to shuffle individual features and observe the impact on model performance. When important features are shuffled, performance typically decreases as the useful relationship between input X and output Y is disrupted. Conversely, shuffling irrelevant features should have little impact or may even improve performance by chance [31].

The algorithm generally follows these steps [24]:

1. Model M is fitted to the training data D (tabular dataset);
2. The reference score s of model M on data D is computed (using the test set or cross-validation);
3. For each feature j (column of D), randomly shuffle the column j of the D dataset to generate a corrupted version of D, then compute the corrupted score s_j of model M on the corrupted D;
4. Repeat step 3 K times for robustness;
5. Importance i_j is defined using the following equation.

$$i_j = s - \frac{1}{K} \sum_{k=1}^{K} s_{j,k} \tag{2}$$

It is essential to note that this equation assumes that higher scores indicate better performance. Therefore, metrics such as R^2 or negative Mean Squared Error (MSE) should be employed. The interpretation of the importance score is as follows: If the shuffled score is lower than the baseline, it suggests that the feature is important, as disrupting the $X-Y$ relationship degraded performance. If the shuffled score is higher, it implies that the feature is less important, as the artificial $X-Y$ relationship improved the results.

Unlike Boruta, which shuffles a copy of the feature, PI shuffles the feature in place. Additionally, Boruta typically employs the random forest model for scoring, whereas PI can utilize any model.

In this work's implementation, PI is applied to individual cross-validation folds. Features are scored within each fold, resulting in five importance values (for five folds). Features are ranked fold-wise, and a median ranking is used as the final relative feature ranking. This approach yields similar results to the standard application, which uses scores only on the test or development set.

3.2.2.2 Sequential Feature Selection

SFS is an interesting feature selection algorithm because of its simplicity. It has no prior assumptions and can be implemented in slightly different manners, but they generally follow these steps:

1. The algorithm starts with 0 features, an empty "basket";
2. Every feature is tested through the model, and the best scoring one (according to a certain metric, such as R^2 or MSE) is added to the "basket";
3. Step 2 is repeated, now combining the tested feature with those already in the "basket". This continues until the basket contains the specified number of features.

SFS is considered a greedy feature selection method and is the "twin" of recursive feature elimination (RFE). RFE works similarly to SFS but starts with a full "basket" and removes features instead of adding them. The main disadvantage of these methods is that while they are simple, they involve a significant number of operations that grow quadratically with the number of features m. With 10 features, 55 operations are necessary; with 10 times more features, 5050 operations are necessary, as depicted by the following equation, m being the number of features and O the number of operations.

$$O = \sum_{k=1}^{m} k = \frac{m(m+1)}{2} \tag{3}$$

In terms of Big O notation, the time complexity of SFS and RFE can be expressed as $O(m^2)$, while other feature selection algorithms, typically have a time complexity of $O(m)$. This means that the number of operations required by SFS and RFE grows quadratically with the number of features, making them computationally expensive for high-dimensional datasets.

3.3 Regression Models

Feature selection algorithms were benchmarked using three different regression models: PLS, Support Vector Machine Regression (SVR), and ANN. PLS regression was selected because it is the single most used model throughout the literature in chemometric applications. SVR is also commonly employed in chemometrics, though not as exhaustively as PLS. Lastly, ANN were selected due to their high flexibility. Given the vast range of possibilities in architecture and hyperparameters, they are deemed able to model relationships with arbitrary complexity given an adequate choice of parameters.

As previously mentioned, proper modeling is crucial for accurately assessing feature selection. Removing features might improve a model's performance, not because those features negatively impact it, but because the model may be overfitted to them. Thus, removing these features helps correct overfitting and enhances the model's performance. Conversely, if the model is underfitted to important variables, it might fail to capture underlying patterns in the data, resulting in poor performance. To address these issues, the following subsections delve into the

inner workings of the employed models and discuss strategies for developing robust models in this study.

3.3.1 Partial Least Squares

PLS is an extension of Multiple Linear Regression (MLR) that is particularly useful for data with correlated, noisy, and numerous features. Unlike MLR, PLS projects both the features and the target variable into a new space. This projection maximizes the covariance between the features and the target while minimizing the difference between the predicted and actual values [32]. In practice, this process creates a model that finds the best possible relationship between the input features and the target variable through the projection. Additionally, PLS is advantageous in situations where the number of features exceeds the number of observations, a common scenario in chemometrics where MLR is unsuitable because it results in a non-invertible covariance matrix. This regression can be represented by the following equation in matrix form:

$$U = TB + H \tag{4}$$

- U represents the target variables projected into the new space. With one target variable, U has dimensions (m), where m is the number of samples;
- T represents the input variables projected into the new space with dimensions $(m \times l)$, where l is the number of orthogonal directions in which the data is projected. These directions are often referred to as the number of components or latent variables;
- B represents the matrix of regression coefficients. With a single target variable, B is a vector of coefficients, with one coefficient for each orthogonal direction l;
- H represents the residual matrix, which is minimized so that the projections are as faithful as possible to the original target and input relationship.

While the exact computation of U and T is very nuanced, practical implementation, as stated before, requires only tuning of the adequate number of orthogonal components l from the T matrix (considered the hyperparameter in this model). The optimal number of components that will yield the best predictive capability varies heavily depending on the dataset. A common and reliable method for selection is through the plot of prediction performance along an increasing number of components, depicted in Figure 3.3. With the increase in orthogonal components, the projected matrix captures more information, improving performance up to a certain point. Beyond this point, the model becomes overfitted to the training data, starting to capture noise, which decreases its ability to accurately predict on new, unseen data (the test set). This plot also represents a cross-validation (CV) score, which serves as an additional metric to ensure that the test set is reserved solely for evaluation purposes. Typically, improvements in CV scores also lead to improvements in the test set performance.

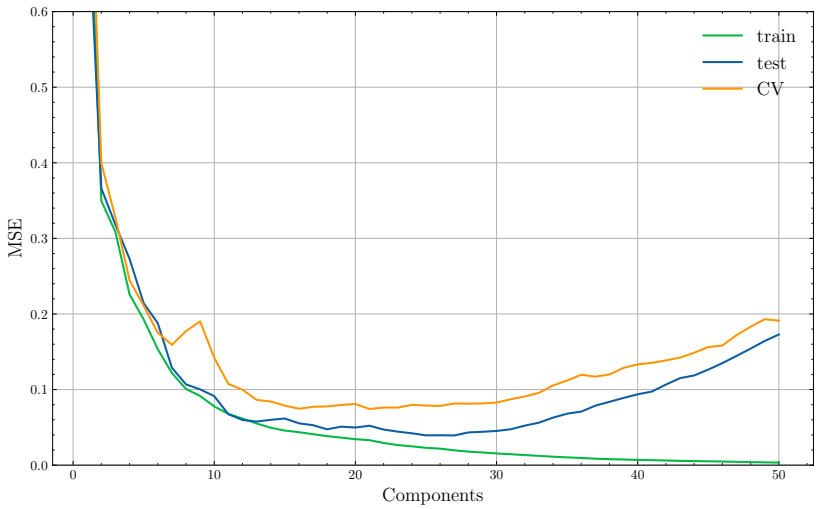


Figure 3.3: PLS's predictive performance over an increasing number of components.

3.3.2 Artificial Neural Networks

Although more cumbersome to implement than PLS and SVR, ANN can be considered conceptually simpler. The basic building block of an Artificial Neural Network is represented by the equation:

$$Y = f(X^T w + b) \tag{5}$$

- Y is a m-dimensional vector, representing the output for each of the m samples;
- X is the $n \times m$ input matrix, where n is the number of features. In spectral data, it is equivalent to the wavelength or wave number. The matrix is transposed so that the dimensions in the equation are coherent;
- w is a n-dimensional weight vector, containing a single value for each of the input features;
- b is a scalar term;
- f is a nonlinear activation function.

In many cases, relationships between data are nonlinear, and thus, modeling capability is limited by only capturing linear relationships. The activation function f is applied to allow the modeling of these nonlinear relationships. Common examples of such functions are the rectified linear unit (ReLU), sigmoid, and hyperbolic tangent. The combination of multivariate linear regression (MLR) with the activation function is referred to as a neuron in the literature.

What allows for the almost arbitrary modeling capability is the use of multiple neurons in multiple layers. This choice of layers and neurons is referred to as architecture. The best-performing structure is specific to each problem (dataset) and is laborious to find because the alternative to seasoned experience is exhaustive search, which is time-consuming and computationally demanding [33].

The way these networks learn the adequate w and b parameters in each neuron to accurately model the problem at hand is depicted by the following steps:

1. Weights and the biases are initialized randomly or by using techniques to estimate a good starting point;
2. A forward pass is conducted to calculate the predictions Y given the initial weights;
3. An error function, such as the mean squared error, is applied to score the difference between the predicted and true values;
4. Using this error, an optimization algorithm, such as the gradient descent, is applied to estimate the direction in which the weights and biases should be changed so that the error decreases;
5. Weights and biases are updated towards the direction in which the error is minimized.

These steps are illustrative of the process of finding a function, composed by weights and a bias, that best transforms observations X into the output Y.

The exact point to stop training may be arbitrary, such as using a specified number of epochs to ensure sufficient performance. A more robust alternative is employing a stopping criterion. These criteria can range from very simple to highly complex. In this study, the criterion applied is as follows: at each epoch, the early stopper checks if the error has improved by more than a predefined delta. If it hasn't, it increments a count and proceeds to the next iteration. When this count reaches a predetermined value, the training stops.

The logic behind this approach is that if the improvement is less than the minimum delta (threshold), the early stopper will wait for a certain number of epochs (often referred to as "patience") for the model to show a significant improvement. If the model fails to do so within the patience limit, the early stopper triggers and halts the training. This process is illustrated in Figure 3.4, a learning curve which represents the error over the learning epochs.

Figure 3.4 depicts the behavior of the stopping criterion. For representative purposes, 6000 epochs are displayed, although the training would have stopped after the yellow vertical line that indicates the triggering of the criterion. This criterion (minimum delta and patience) is tuned to trigger when the data is neither overfitted nor underfitted, as seen in the plot. Continuing the training beyond the yellow line only improves predicted performance slightly, and shortly after, improvements halt and overfitting to the training data starts to occur.

When assessing feature selection, it is essential to use a stopping criterion, since selecting subsets of features can significantly change the learning curves of the model. With a reduced set of features, the model might reach convergence (the point at which the model is adequately fitted) earlier or later than it would when using the full set of features. Therefore, using a fixed number of epochs for training, may not be suitable when working with different subsets of data. The stopping criterion helps to ensure that the model is trained for an appropriate number of epochs, regardless of the specific features being used. A similar, though less pronounced, behavior can also be observed in the different folds of cross-validation.

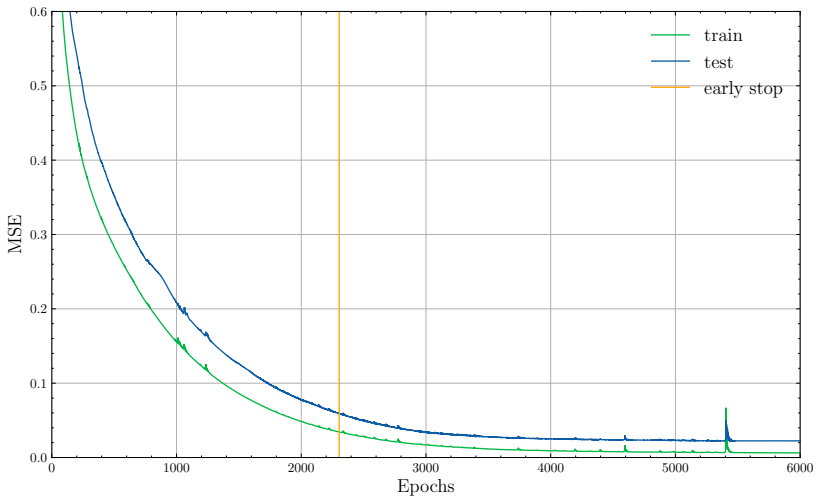


Figure 3.4: Learning curve, Dataset 1.

Figure 3.5 illustrates the behavior described above when the same model is trained on a subset of the data, a situation that always occurs with feature selection. With the original dataset, about 2300 epochs are enough to reach a mean squared error of 0.05 for predictions, while this same metric is achieved on the subset trained with 1200 epochs. Stagnation and overfitting also start occurring earlier when training on the subset. This can be attributed to several factors, but is most likely related to the fact that a model with fewer features is simpler, has fewer parameters, and thus is "easier" to fit to the data.

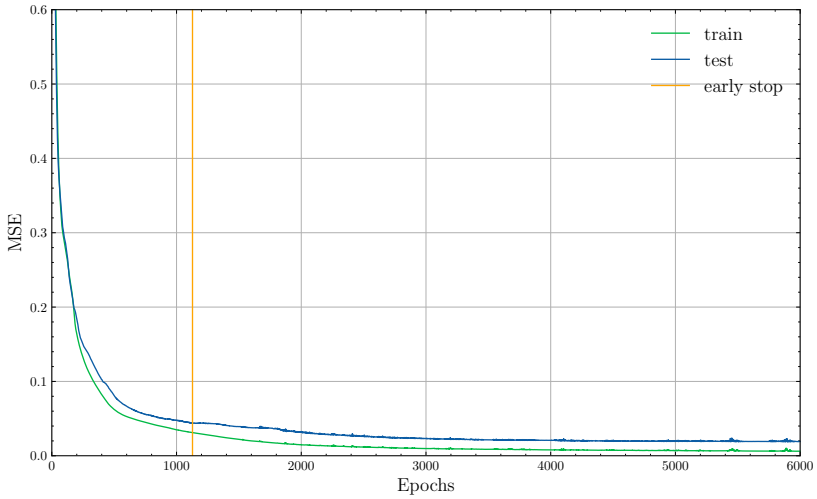


Figure 3.5: Learning curve where 201 less important features were removed by MI, Dataset 1.

An important point is the use of the Adam optimization algorithm, which differs from simple gradient descent in that it adjusts the learning rate dynamically. While Adam speeds up training, it can sometimes increase the error during certain epochs. This isn't usually an issue, as the error typically decreases more in subsequent iterations. However, in the specific context where a stopping criterion is employed, there is an unlikely but possible case in which the stopper triggers on a "peak" resulting in a poorly performing model. A fail-safe was implemented that

overrides the patience counting. If the error is higher than the error of the last epoch, even if it is inside the minimum delta range, this event is ignored and training continues, avoiding improper stopping.

A final aspect that was not tackled earlier relates to the manner in which networks learn. Instead of updating weights only when all data points are included in the error calculation, they are frequently updated in batches, as this approach offers advantages in terms of efficiency and stability [24]. However, it was observed that by using smaller batch sizes "peaks" during the training process appear more frequently, which increased the risk of the stopping criteria being triggered improperly. Thus, larger batch sizes were favored to ensure that stopping would always occur appropriately, preventing any undesirable influence on the perception of feature selection.

3.3.3 Support Vector Machines

Lastly, support vector machines are applied because their use in chemometric applications has been increasing in recent years, owing to their ability to handle high-dimensional data and robustness to overfitting [34].

They work by finding the best line (or hyperplane) within a certain threshold of the target variables. Their flexibility is derived from the use of kernel functions, which enable the algorithm to perform effectively in high-dimensional spaces by implicitly mapping the input features into higher-dimensional feature spaces. Two common kernels used in chemometrics are the linear kernel and the radial basis function (RBF) kernel [34].

The selection of these well-established and commonly employed methods is considered sufficient for assessing the general behavior of feature selection.

3.4 Convolutional Neural Networks

As previously mentioned, CNN are used as an alternative to the combined use of feature selection and traditional regression models. CNN have shown exceptional success in image processing tasks, and recent studies suggest that they could also be effective as regression models in chemometrics [3].

The fundamental difference between CNN and ANN is that CNN include at least one convolutional or pooling layer in addition to the standard dense layers (neuron layers) found in ANN [35].

The convolution operation can be understood as taking a weighted moving average of a time-dependent continuous variable, such as the position as a function of time. This is represented by the following equation [35]:

$$s(t) = \int x(a)w(t-a)\,da \tag{6}$$

- $s(t)$ is the result of the convolution at time t;
- $x(a)$ is the input function, representing the position at time a;

- $w(t-a)$ is the weighting function (or kernel) that shifts with respect to t.

This results in, $s(t)$ which is a smoothed estimate of the position at time t, considering contributions from past positions $x(a)$ weighted by $w(t-a)$. This operation, called convolution, is commonly denoted with an asterisk [35]:

$$s(t) = (x * w)(t) \tag{7}$$

In the specific terminology of CNN, the function uses the input $x(a)$, and a window function $w(t–a)$ or w, which is often referred to as a kernel or filter. The resulting output is sometimes called a feature map [35].

The key concept behind convolutions is that, similar to a moving average, it involves sliding a kernel over the input data to produce a feature map. This operation is akin to a cross-correlation, which involves computing the dot product between the kernel and the input at each position [35].

Convolutions, while mathematically complex, are visually straightforward. This is illustrated in Figure 3.6, which shows how a fixed kernel is applied to a 2D image. The figure is adapted from [36].

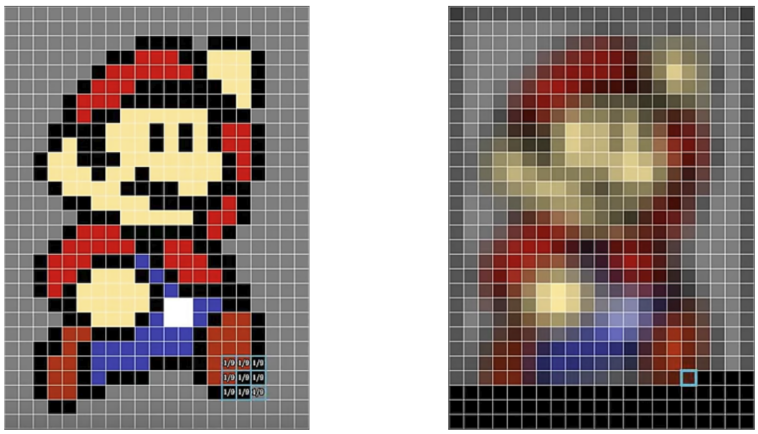


Figure 3.6: Visual intuition on convolutions, adapted from [36].

What occurs in the example is that a 3×3 grid of values (kernel) slides on top of the input pixels, row by row, resulting in the output feature map. At each iteration, each value is multiplied by the corresponding pixel it sits on top of, and when added, it results in an average of each color channel. In CNN terminology, the image on the right is said to be the result of a convolution of the image on the left with the specified kernel, as depicted in Figure 3.7, adapted from [36]. A mathematical technicality is that the kernel is actually rotated 180 degrees clockwise; in this specific example, since the kernel is symmetrical, the result is indifferent [35].

While in this specific example, the convolution only blurs out the initial input, its actual capabilities go much further. The actual weight values in the kernel change in each learning iteration (epoch); they are optimized to minimize the error, the same way the weights w of the MLR are optimized in the neuron of ANN. Analogous to the MLR, convolutions involve a bias b that was omitted for simplicity.

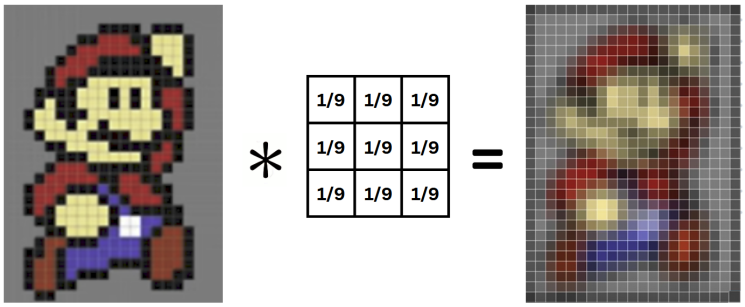


Figure 3.7: Pseudo-equation representing the convolution operation, adapted from [36].

The primary difference between the illustrative 2D image processing and spectral image processing is that spectra such as NIR, Raman, and FTIR are one-dimensional. Consequently, the kernel used in spectral image processing is also one-dimensional. The concept of "sliding" the kernel across the data, as done in 2D image processing, is similarly applied in one-dimensional spectral data, but only along a single row.

Additionally, the convolution operation in spectral image processing is somewhat akin to the widely used Savitzky-Golay (SG) filter. However, there are key differences in both the mathematical operation and parameter determination. The convolution kernel's parameters are learned from the data, while the SG filter uses predetermined parameters. In terms of the mathematical operation, convolution involves a weighted sum of the input data, whereas the SG filter applies a polynomial smoothing method. In summary, convolution optimizes the processing method based on the data itself, whereas the SG filter follows a fixed approach.

Convolutions leverage some important concepts that can help improve a machine learning system: parameter sharing and sparse interactions. Traditional Neural Network layers use matrix multiplication, a vector for each neuron with a distinct parameter for each input unit and output unit, while CNN employ a kernel that is smaller than the input [35]. As depicted in Figure 3.8, taken from [35], in the standard ANN, the input x_3 affects all the s outputs since s is formed by matrix multiplication; in the convolution, x_3 only affects s_2, s_3, and s_4 due to the limited kernel window.

Additionally, traditional neural networks assume that data points are independently and identically distributed (i.i.d). This assumption does not hold for Convolutional Neural Networks (CNN), which makes them particularly suited for tasks like image processing and spectral data analysis, where data points are correlated. In these cases, the assumption of i.i.d. does not apply, and CNN can effectively exploit the local structure in the data. This makes the application of CNN to spectral data more sensible, as spectral data is typically not i.i.d.

The advantage is that by using the same weights for all the inputs and limiting the connection range by kernel size, convolutions can reduce complexity by reducing the total number of parameters. This propriety contributed to making CNN the standard choice for tasks that involve many inputs, such as image processing [35].

The other operation employed in CNN is the pooling operation. Similar to convolutions, pooling takes a filter (kernel) and slides the filter along the input generating a feature map, but unlike convolutions, pooling does not involve weights. Instead, it employs a mathematical operation,

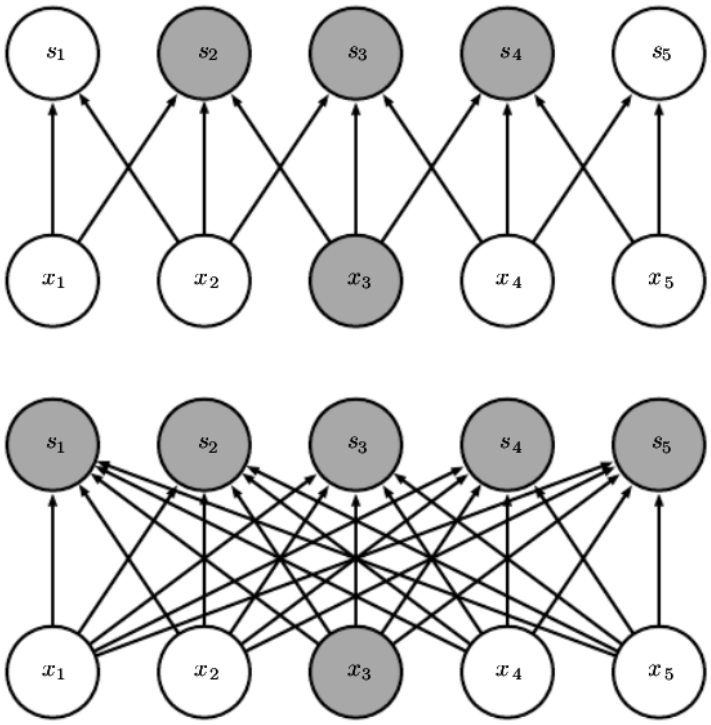


Figure 3.8: Sparse connectivity example: convolution on top, dense layer below, taken from Goodfellow et al. (2016) [35].

such as taking an average or maximum value, to generate a feature map [35]. Pooling is very useful to reduce the total number of layers, which is very useful in deep learning applications, but for spectral data processing literature suggests that its utility is limited [3].

Two important nuances that are yet to be referred to are the terms “channel” and “stride”. Stride refers to the step size of the kernel as it moves across the input; a larger stride means the kernel skips more input elements, resulting in a smaller feature map. Figure 3.6 depicts a stride of 1; with a stride of 2, the filter would move two steps at a time and so on. Channels refer to the different dimensions in which inputs or outputs may be stacked. In Figure 3.6, each colored pixel actually represents a vector of 3 values: red, green, and blue, corresponding to the three color channels. In the context of spectral image processing, inputs are typically one-dimensional and single-channel, but multichannel kernels can be used to produce a multichannel feature map, capturing new relationships within the data.

Computing the exact size of the feature map is necessary for the application of CNN; thus, the following equation is used.

$$\text{Feature Map Size} = \left\lfloor \frac{N - F + 2P}{S} + 1 \right\rfloor \tag{8}$$

- N represents the number of points in the spectra, the included wavelengths or wavenumbers;
- F represents the kernel size;
- P represents the padding;

- S represents the stride.

It is important to note that Equation 8 employs floor division. Padding refers to additional virtual sample points that may be added to the left and rightmost points of the spectra, the edges. Among many works, padding was never used for predictive modeling of spectral data.

Although CNN differs from ANN due to parameter sharing and sparse interactions, they inherit many of the modeling techniques and parameters used in standard neural networks such as the batch size, optimization algorithm, and loss function. Adam and mean squared error are also common choices for CNN [3].

Much alike the ANN, CNN also inherit the architecture search difficulty that ANN has. The optimal parameters and architecture are specific to each problem and laborious to find. Thus, the specific architecture employed by Zhang et al. (2019) [21] is employed in this work because it produced results that are consistent and superior to the standard PLS and SVM models even on distinct datasets.

The specific architecture employed by Zhang et al. (2019) [21] utilizes a so-called Inception module, introduced by Szegedy et al. (2015) [37] as part of the bigger GoogLeNet. This utilizes convolutional layers, with different kernel sizes in branches, as depicted in Figure 3.9, adapted from [37] and [38].

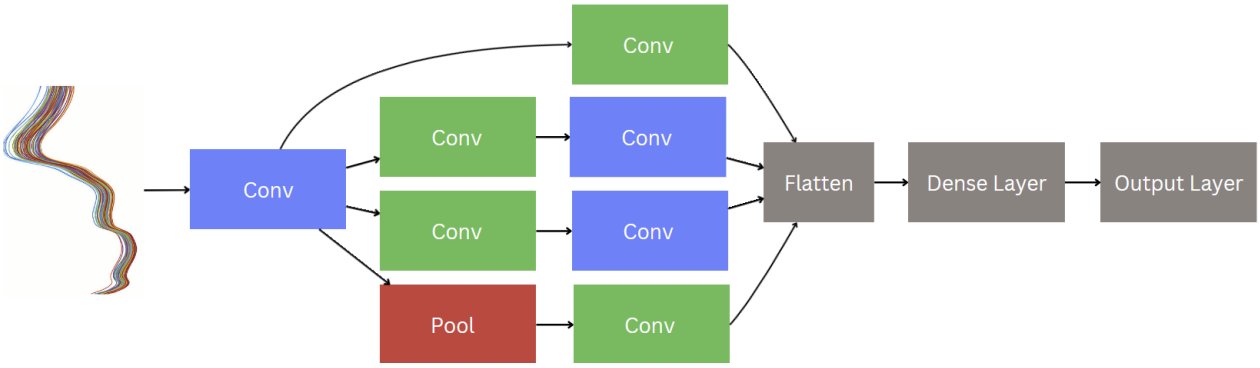


Figure 3.9: Representation of the architecture with the Inception module, adapted from Szegedy et al. (2015) [37] and Yan et al. (2019) [38].

The image illustrates the Inception module, where green represents a convolution with a kernel size and stride of one, blue indicates a convolution with varying kernel sizes (differing between blue modules), and red denotes a pooling layer.

The reasoning behind the use of this specific structure identified by Szegedy et al. (2015) [37] is:

- Multiple filter sizes: The Inception module uses filters of different sizes (for example 1, 2, and 3) within the same module. This allows the network to capture features at different scales and spatial resolutions;

- Parallel processing: The different-sized filters are applied in parallel and their outputs are concatenated. This allows the network to learn multi-level feature representations efficiently;

- Feature aggregation: The output of an Inception module is a rich, multiscale feature representation that combines information from different spatial resolutions. The smaller kernels capture very fine-grained, localized features, while the bigger kernel size convolutions capture progressively coarser, more global features.

The strong empirical results achieved by GoogLeNet, which employed the Inception architecture, supported the effectiveness of this design choice.

The architecture, proposed by Zhang et al. (2019) [21], also employed the following techniques to improve predictive modeling:

- **Batch normalization**: A technique that normalizes feature maps or inputs inside the layers of the network. This is generally applied to speed up the learning phase;
- **Dropout regularization**: A technique that randomly zeros some outputs inside the network, making the model less reliant on specific parameters and often effectively reducing overfitting;
- **Variance initialization**: A technique employed to avoid exploding or vanishing gradients that may occasionally happen in larger networks. This method is used to select robust initial values that are neither too large nor too small.

4 Results and Discussion

This chapter evaluates four feature selection algorithms—mRMR, MI, PI, and SFS—using ANN, PLS, and SVR models on two datasets. It discusses the effectiveness of these methods in improving model accuracy and reducing dimensionality, along with key settings, performance metrics, and the behavior of each model with different feature selection strategies. Additionally, CNN were assessed as an alternative modeling technique.

4.1 Feature Selection

To evaluate the general behavior of feature selection, three models—ANN, PLS, and SVR—were applied using four different algorithms: mRMR, MI, PI, and SFS to the two datasets.

Two distinct types of tests were conducted: non-exhaustive search, where model hyperparameters remained constant throughout the feature selection process and exhaustive search, where the model's hyperparameters were optimized simultaneously with feature selection.

4.1.1 Preprocessing

Preprocessing for Dataset 1 involved the standard normalization of both the target variable Y and input features X. The dataset was then randomly split into training and testing sets with an 8:2 ratio.

Dataset 2 underwent the same 8:2 split ratio and the preprocessing detailed in the source of the dataset [14]. Similar ratios yielded comparable results, and this specific choice is a common option in predictive modeling.

During the modeling of Dataset 2, a significant imbalance problem was observed. Most of the germination rate samples had high values. With a random split, it was possible that most of the high-valued samples would be allocated to the training set while the low values would be allocated to the test set. In such cases, the information that accurately predicts high rates may differ from those adequate for predicting low values, potentially leading to poor model performance. Ideally, when randomly split, the sets should generally be independent and identically distributed (i.i.d.), which was not the case.

To mitigate this issue, the samples were sorted based on the germination rate (target variable) and then sequentially allocated to the training and test sets. This allocation method ensures that the distribution of samples between the training and test sets is similar, mitigating the imbalance problem.

The imbalance problem from Dataset 2 also affects cross-validation. To mitigate this issue, repeated cross-validation was applied. When the training data is randomly split between folds, the distribution might be more or less balanced in some splits, and the average performance between splits might not be very representative of the actual model capability. To better estimate the model's performance, cross-validation was repeated 5 times with different random allocations between folds, resulting in 5 different CV that, when averaged, are better representative of the expected performance. This final metric is considered more robust and better

representative of the model's behavior.

4.1.2 Modeling

As previously discussed, overfitting or underfitting can undesirably affect the evaluation of feature selection methods. Thus, robust models are crucial for accurately assessing feature selection performances. For PLS and SVR, model development is straightforward; selecting an appropriate number of latent variables for PLS and the kernel for SVR are sufficient to yield models with adequate performances, as presented in Table 4.1. Notably, the choices of 16 and 20 latent variables for PLS gave the best CV values with the full set of features. In contrast, ANN development is more challenging due to the complexity of its architecture and the high number of hyperparameters.

Table 4.1: Dataset comparison.

	Dataset 1	**Dataset 2**
PLS # latent variables	20	16
SVR kernel	linear	linear

The ANN architectures and hyperparameters listed in Table 4.2 were thoroughly benchmarked with different initializations, consistently demonstrating robust performance across multiple evaluations, with results summarized in Table 4.3.

Table 4.2: Comparison of ANN configurations.

	Dataset 1	**Dataset 2**
Architecture	Layer 1, 4 nodes Layer 2, 4 nodes Output, 1 node	Layer 1, 6 nodes Output, 1 node
Activation function	Layer 1, ReLU Layer 2, ReLU Output, None	Layer 1, tanh Output, sigmoid
Opt. Alg.	Adam	Adam
Loss function	Mean squared error	Mean squared error
Learning rate	0.001	0.001
Batch size	316	395
Patience	25	200
Min. Delta	0.001	0.00005

The ANN architectures and hyperparameters summarized in Table 4.2 were selected to enhance the predictive performance while prioritizing commonly used techniques such as the ReLU activation function and Adam optimizer [39, 40]. Hidden layers are referred to as "Layer" and the input layer is omitted. When applicable, the settings used by Ma et al. (2024) [14] were implemented.

Regularization techniques were not analyzed because they did not meaningfully impact the performance and could potentially influence feature selection. The same reasoning applies to avoiding additional SVR hyperparameters such as C and ε.

Table 4.3 summarizes the performance metrics for each model on both datasets using the full set of features, which serves as a baseline. CV scores are presented as *meanCV(std)*, where the value in parentheses is a convenient way to represent the standard deviation. For example, 0.068(8) represents 0.068 ± 0.008. Overall, the model performances are consistent, with low values of mean squared error (MSE) and R^2 values close to one, except for SVR on Dataset 2 and the worse CV scores of the ANN on Dataset 2.

Table 4.3: Model performance on Dataset 1 and Dataset 2.

		MSE			R^2		
Dataset	**Model**	**Train**	**Test**	**CV**	**Train**	**Test**	**CV**
1	**PLS**	0.034	0.050	0.068(7)	0.966	0.946	0.929(18)
	SVR	0.040	0.052	0.067(7)	0.960	0.943	0.931(11)
	ANN	0.044	0.071	0.075(20)	0.955	0.923	0.923(26)
2	**PLS**	0.013	0.033	0.037(8)	0.877	0.704	0.642(110)
	SVR	0.047	0.047	0.051(7)	0.557	0.576	0.504(90)
	ANN	0.031	0.033	0.068(37)	0.705	0.699	0.354(317)

The lower performance of SVR on Dataset 2 could be due to the dimensionality and distribution of the dataset. Dataset 2 contains 1846 features, approximately 4.6 times more than Dataset 1, which suggests that Dataset 2 may have a more challenging relationship between X and Y, as well as more redundant information or noise that negatively affects performance. Additionally, the imbalanced nature of Dataset 2, characterized by a skewed distribution with a higher frequency of high Y values, further complicates the modeling process. This imbalance is evidenced by the high standard deviation of the CV scores. To address this, repeated cross-validation was employed. While this technique improved the performance of PLS, only minor improvements were observed on the other models.

The worse CV scores of the ANN on Dataset 2 can be derived from its higher sensitivity to the imbalance problem compared with the other models. ANN training involves more complex mechanisms, such as a stopping criterion, which can fail on some folds and trigger improperly, resulting in poor scores. This is emphasized by the standard deviation being nearly equal to the mean. Repeated cross-validation was employed to mitigate this issue, and the results for Dataset 2, as shown in Figure 4.3, already reflect this repeated cross-validation with 5 repetitions to obtain better estimates of mean and standard deviation values. Despite these efforts, the repeated cross-validation was only marginally successful at a significant computational cost owing to the longer training times of ANN compared to the other models. It is uncertain whether the model parameters or the dataset imbalance is the primary cause of this fold-level failure; it is likely that both factors are equally responsible. Nonetheless, the selected models were considered sufficient for evaluating the feature selection methods.

Finally, it is worth noting that slightly better performance could be achieved by training the

ANN for longer, particularly on Dataset 1. However, for simplicity, a more conservative stopping criterion was used to avoid overfitting.

4.1.3 Filter Methods

Filter methods, as discussed in Chapter 3, assess feature relevance independently of the learning algorithm, typically by ranking the features based on statistical measures. In this study, two filter methods were selected for evaluation: mRMR and MI. mRMR was chosen for its ability to select features that are maximally relevant to the target while being minimally redundant. MI was included for comparison because of its effectiveness in capturing non-linear relationships between features and the target.

The application of filter methods yielded promising results in dimensionality reduction while maintaining model performance. Both mRMR and MI were able to significantly reduce the number of features used in the models, in some cases by over 60 %, with only minor impacts on predictive accuracy. This strongly suggests that a large portion of the original feature set is redundant or irrelevant to the prediction task. Interestingly, the two methods exhibited fairly similar performance, with barely any advantage of one over the other, apart from MI's theoretical advantage in capturing nonlinear relationships and its faster runtime.

The tests performed are straightforward: the model is trained with different subsets of data and scored, and the results are plotted for visualization. These results are presented and discussed in the following sections.

4.1.3.1 Minimum Redundancy Maximum Relevance

Figure 4.1 depicts the performance of PLS with a fixed number of latent variables (20) when using features ranked by mRMR. The bars in the cross-validation (CV) score represent the standard deviation of the scores across folds. The general trend shows that performance changes only slightly, with the CV score worsening slightly, while the train and test scores remain relatively constant, until approximately 75 % of the total features are removed. Further removal leads to a significant decrease in performance, likely because the algorithm has stopped removing redundant information and is now removing features important for target prediction. The choice of 20 latent variables was made because it yielded the lowest cross-validation (CV) error, ensuring optimal model performance. For Dataset 2, 16 latent variables are used, as this number provided the lowest CV error for that dataset.

Figure 4.2 presents the same test but with a lower number of PLS latent variables. The main difference is observed in the model with only 10 latent variables, where a significant increase in performance is seen, especially in the CV metric. This could possibly be attributed to the limited modeling capability with a low number of latent variables, as the PLS struggles to successfully extract new features from the original set. However, by removing some features, the 10 latent variables become sufficient to correctly model the underlying relationships.

Figure 4.3 illustrates the behavior when using the best-performing number of latent variables on the CV score for each subset of features. Compared to the other tests, this one is relatively

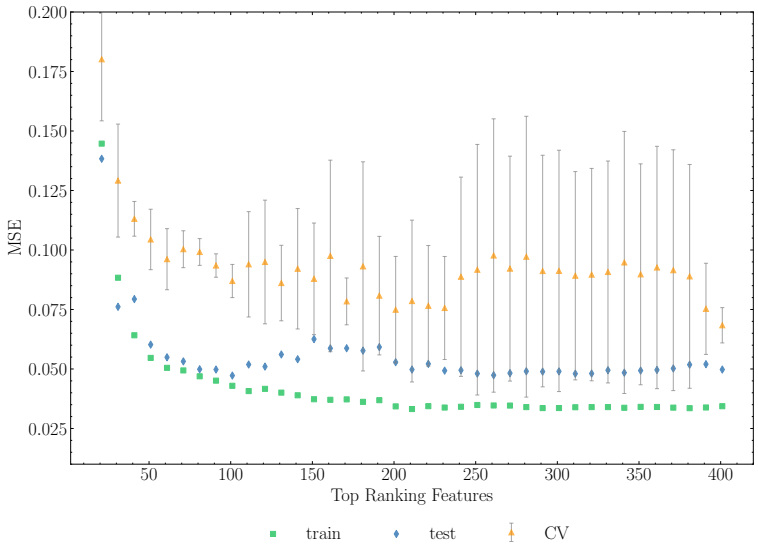
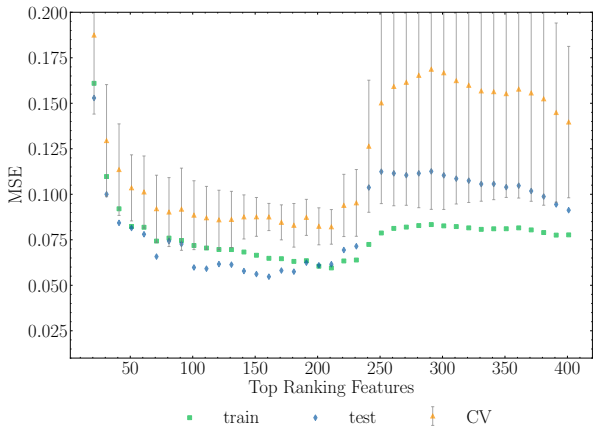
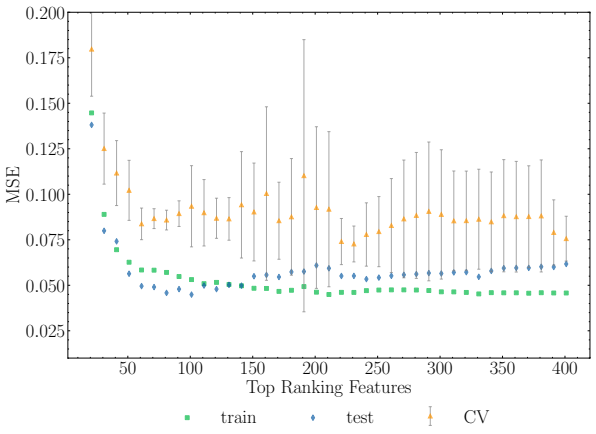


Figure 4.1: PLS MSE with 20 latent variables over features ranked by mRMR, Dataset 1.



(a) 10 latent variables.



(b) 15 latent variables.

Figure 4.2: PLS MSE with a lower number of latent variables over features ranked by mRMR, Dataset 1.

more stable, with a significantly smaller standard deviation in the CV score, likely because the model has the most adequate number of latent variables for modeling each specific subset. Even with this exhaustive search, there is a slight worsening in performance, but a significant number of features can still be removed. The performance degradation threshold seems to be almost global, as no number of latent variables can successfully model the X-Y relationships after removing approximately 85 % of the total features.

Additionally, Figure 4.4 shows the optimal number of latent variables used in Figure 4.3. This behavior aligns with general intuition: fewer variables generally imply a lower ideal number of latent variables, even if a smaller subset of variables is potentially more informative.

The ANN, shown in Figure 4.5, exhibits a more unstable behavior compared to PLS, which is expected because CNN can be trained for an arbitrary number of epochs, solely controlled by the triggering of the stopping criteria in this specific test. An investigation into the stopping criteria revealed that the subsets with the worst model performance had, on average, significantly fewer

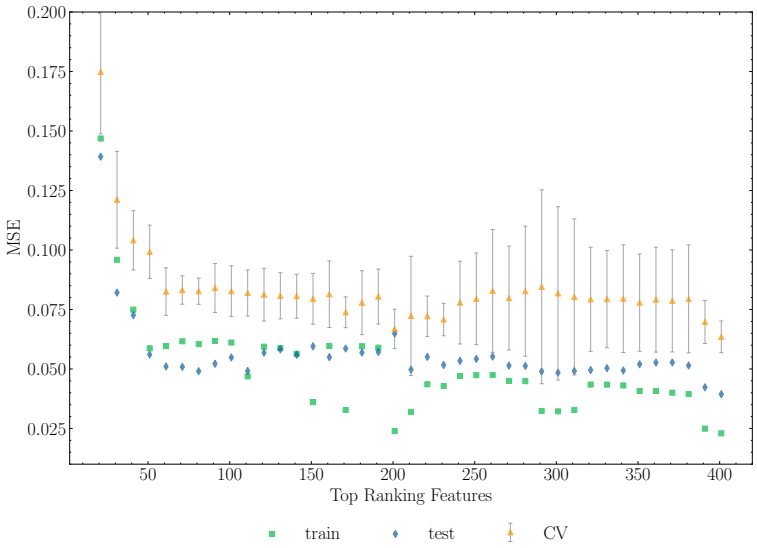


Figure 4.3: PLS MSE with the best number of latent variables over features ranked by mRMR, Dataset 1.

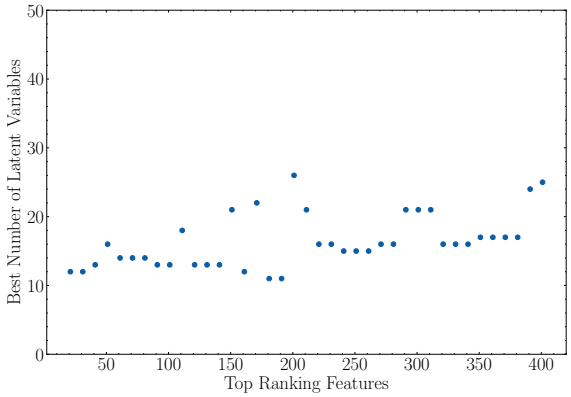


Figure 4.4: Best number of latent variables over features ranked by mRMR, Dataset 1.

epochs (Table A.1), suggesting underfitting in these iterations. This observation essentially confirms the same trend observed in the other models: removing the lowest-ranked features allows the model to maintain similar performance up to a certain point, beyond which the performance deteriorates. Some instances where the performance surpasses that of the full set can also be observed, which may be attributed to the inconsistency of the stopping criterion across different data subsets, allowing more epochs for the model to reduce its error.

Finally, the SVR in Figure 4.6 is significantly more stable, with a much clearer trend than PLS and ANN. This observation aligns with the general understanding that SVR tends to be more stable and less prone to overfitting compared to PLS and ANN models. SVR's ability to find a hyperplane that maximizes the margin between data points helps in achieving better generalization and stability.

For Dataset 2, the trends are generally very similar regardless of the chosen model. For variation and distinction from Dataset 1, the results for Dataset 2 are displayed using the R^2 metric instead of MSE. The trends are similar but essentially flipped, as a good R^2 score is close to one, while a good MSE score is close to zero.

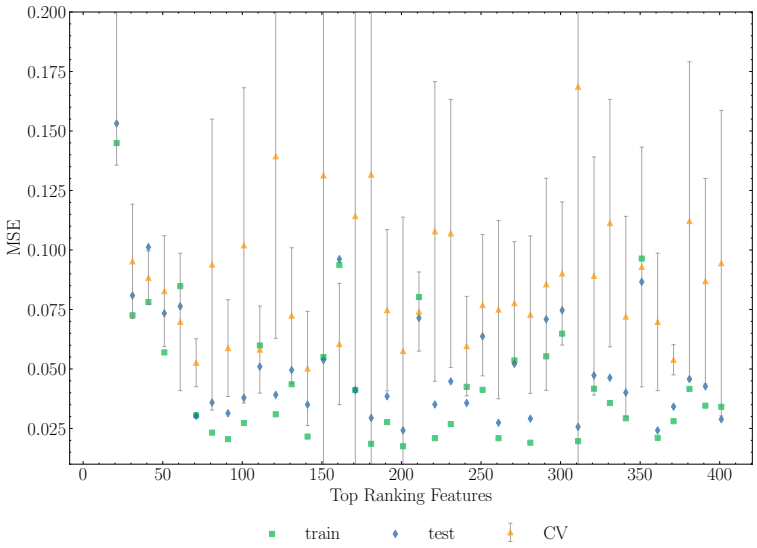


Figure 4.5: ANN MSE over features ranked by mRMR, Dataset 1.

Figure 4.7 depicts PLS with an exhaustive search for the best number of latent variables for each subset of features. The behavior is similar to that of Dataset 1, where the performance fluctuates around values close to the starting point until a threshold of features is reached, after which the performance starts to decrease. The threshold was slightly higher than that in Dataset 1. Both the standard deviation and other scores were more stable than those in Dataset 1, which could be attributed to the fact that Dataset 2 contains significantly more features.

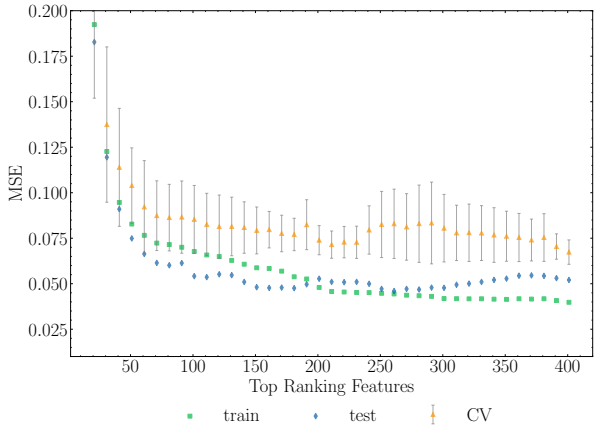


Figure 4.6: SVM MSE over features ranked by mRMR, Dataset 1.

Figure 4.7 also shows a decreasing trend in the training set, suggesting that less overfitting occurs with fewer features. However, this trend is more likely to occur because, with fewer features, the ideal number of latent variables is lower, and thus, the fitting capability also decreases. This is very likely because this trend was only observed in the exhaustive test.

Figure 4.8 presents the PLS with 16 latent variables and SVR. The main conclusion is that fixing the number of latent variables diminishes the number of total features that can be removed, most likely because using numerous latent variables with a not-so-large number of features contributes to a high fitting capability that leads to overfitting. SVR has very stable results, but considerably

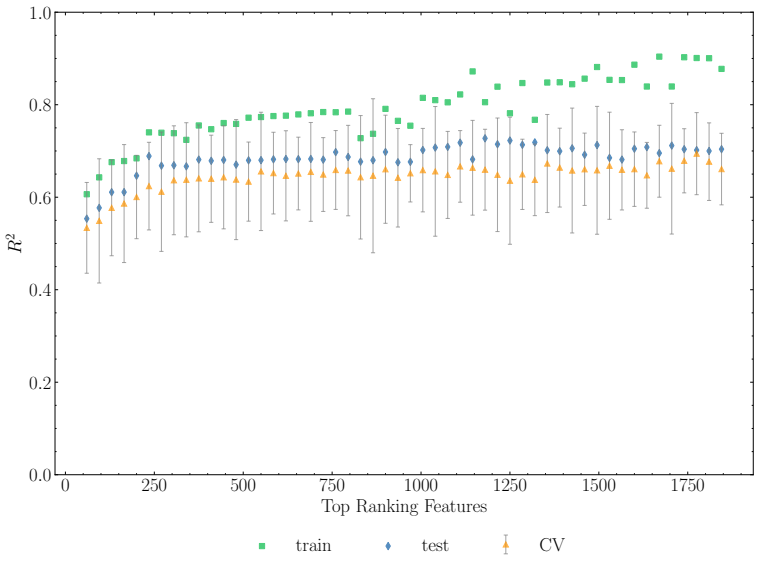
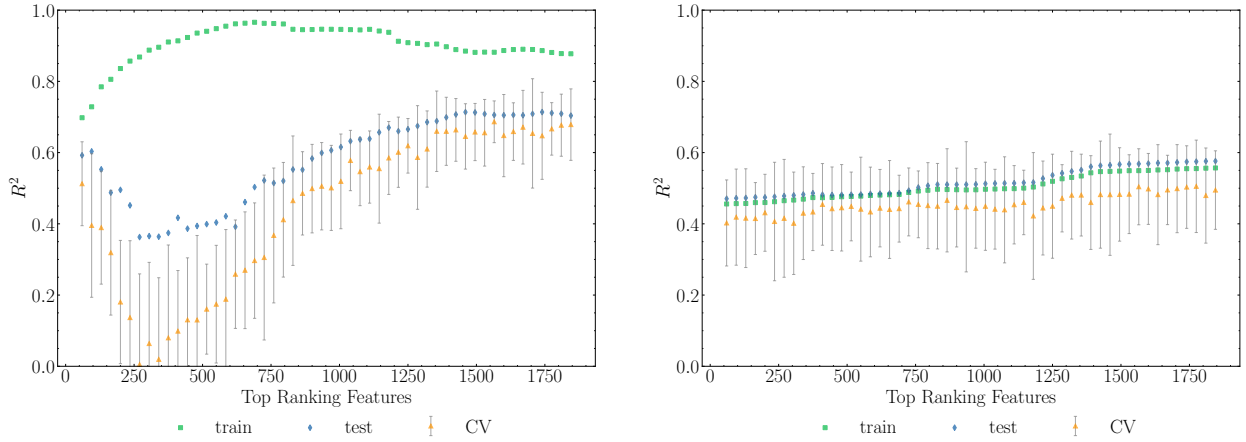


Figure 4.7: PLS R^2 with the best number of latent variables over features ranked by mRMR, Dataset 2.



(a) PLS 16 latent variables.

(b) SVR linear kernel.

Figure 4.8: PLS and SVR R^2 over the best features ranked by mRMR, Dataset 2.

lower accuracy than the other models. This could either be attributed to the nature of SVR or the fact that with low accuracy, it could mean that the SVR does not have much success in drawing relationships between the data, and feature selection does not alter this.

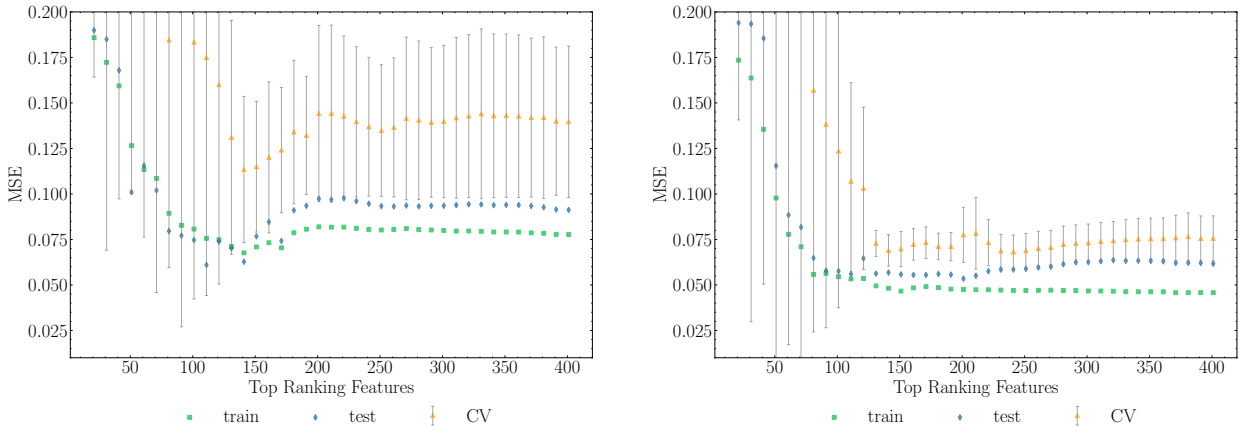
The general conclusion is that no model with fixed or best parameters achieved higher accuracy with a subset of the original features than with the full set. However, a very high ratio of total features can be removed without significantly affecting the predictive accuracy.

4.1.3.2 Mutual Information

One glaring difference between mRMR and MI, which is evident even before any result is observed, is the runtime. On smaller datasets, mRMR takes seconds while MI takes milliseconds; this speed difference is even more evident in larger datasets with more numerous features, such as Dataset 2, where on this specific machine, mRMR takes on average 3 minutes while MI takes 3

seconds.

By directly comparing mRMR and MI with a PLS with a fixed number of latent variables, Figure 4.2 and Figure 4.9, it is evident that MI is slightly more successful in selecting features usable by PLS, especially with 15 latent variables. The plots are considerably smoother with a lower standard deviation, indicating a better X-Y relationship. This could be due to the fact that mRMR scoring seldom works precisely for the specified datasets, or possibly the fact that features informative to the target variable generally help the model to predict the target variable, even if these features have redundant information, which aligns exactly with the inner workings of MI that only score how informative features are, with no regard for redundancy.



(a) 10 latent variables.

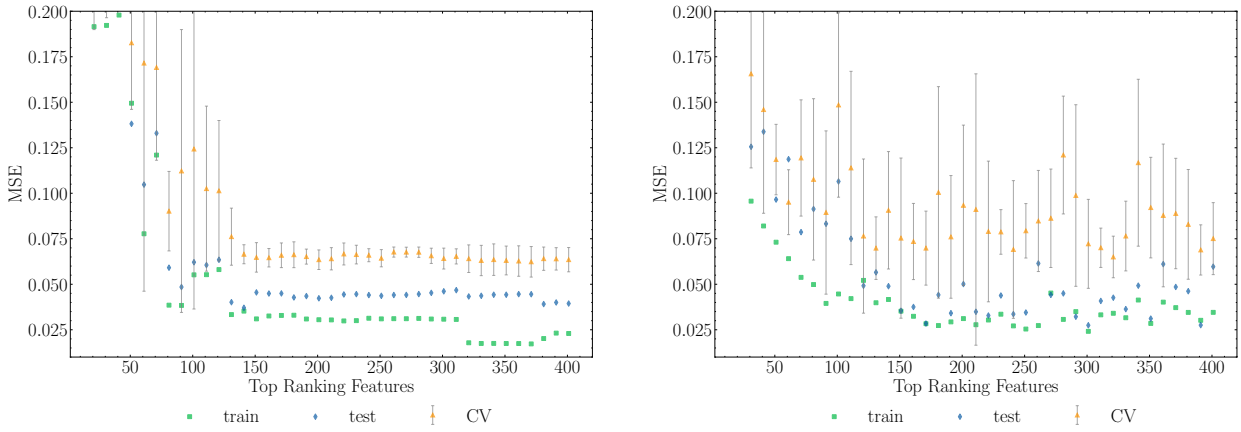
(b) 15 latent variables.

Figure 4.9: PLS MSE over features ranked by MI, Dataset 1.

Both the exhaustive PLS test, with the most adequate number of latent variables on each subset of the data, and the ANN test (Figure 4.10) achieved more stable curves than those obtained with mRMR. This could mean that the models successfully extract information from the features, regardless of the redundancy; thus, scoring redundancy in the features negatively impacts the behavior, or that Dataset 1 does not have enough redundancy, and thus, the use of mRMR is irrelevant. The latter does not seem to be the case because the trends are analogous, regardless of the dataset. Figures containing the MI algorithm on Dataset 2 are presented in Figure A.1. Instability in the neural network can analogously be attributed to improper triggering of the stopping criterion; however, the trends were smoother with the MI algorithm.

4.1.4 Wrapper Methods

As discussed in Chapter 3, wrapper methods employ a specified model to score features. This has theoretical advantages and disadvantages compared with filter methods. Using the model in question to score features guarantees that the features scored by the model have information useful to the model at hand. In contrast, when filter methods score features, they score the features highly informative of the target variable, but it is possible that this information is unusable by the model in hand. The disadvantage is that wrapper methods generally "lock" the model parameters since they use the model itself to score features the parameters have to



(a) PLS.

(b) ANN.

Figure 4.10: PLS with the best number of latent variables and ANN over features ranked by MI, Dataset 1.

be determined beforehand. Exhaustive tests, in which parameters and features are searched simultaneously, are only sensible for filter methods.

4.1.4.1 Permutation Importance

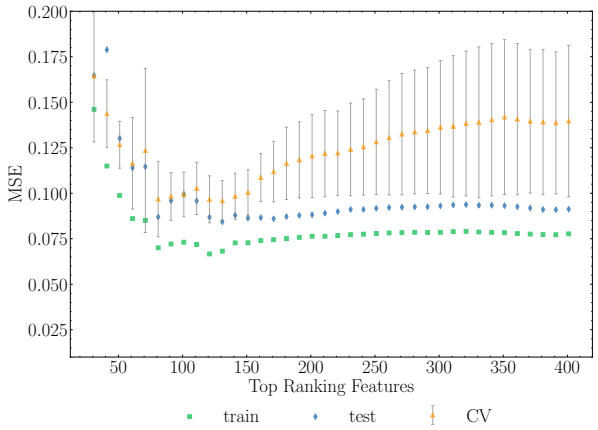
Unlike filter methods, which rely solely on data to score relationships, PI is a wrapper method that scores features using a specified model. This fundamental difference introduces a risk: if the same set is used for both generating the model and scoring features, the highest-ranked features may be overly tailored to that specific set, thereby impairing generalization and potentially leading to overfitting.

To mitigate this risk, a specific cross-validation algorithm was applied. The model was trained on the training folds, while feature selection was performed on the validation folds. The median ranking of each feature across these folds was then considered for the global ranking. Feature selection is then conducted using this global ranking.

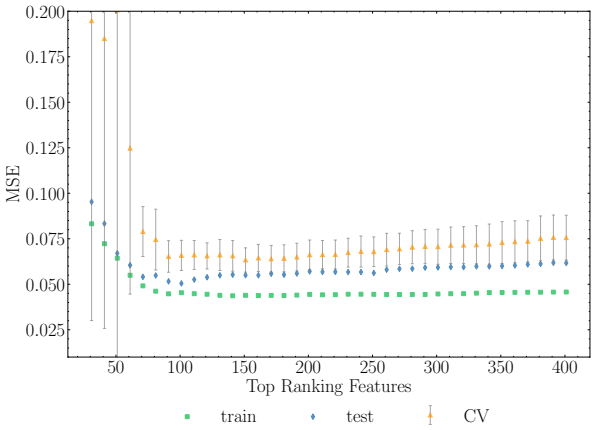
Figure 4.11 shows a well-defined downward trend for the errors, indicating that removing lower-scored features slightly improves performance with each feature removed. This trend holds true, regardless of the number of latent variables used in the PLS model.

However, in Figure 4.12, which uses 20 latent variables and yields the best CV with the full set, this trend is weaker. Although the improvement was more pronounced with a lower number of latent variables, it did not surpass the results achieved by the model with a higher number of latent variables. With 25 latent variables, as shown in Figure 4.12, the model slightly overfits owing to the excessive number of latent variables. Interestingly, the model with 25 latent variables performed better on the test set at some iterations, but worse on the CV score, suggesting a potential imbalance problem in the dataset.

Figure 4.13 shows the behavior of the neural network when features are selected by PI on Dataset 1. Compared with mRMR and MI, it is clear that the PI algorithm produces a smoother curve. This is likely because PI chooses features informative to the target that are usable to the model at hand, which aligns with its theoretical intuition.

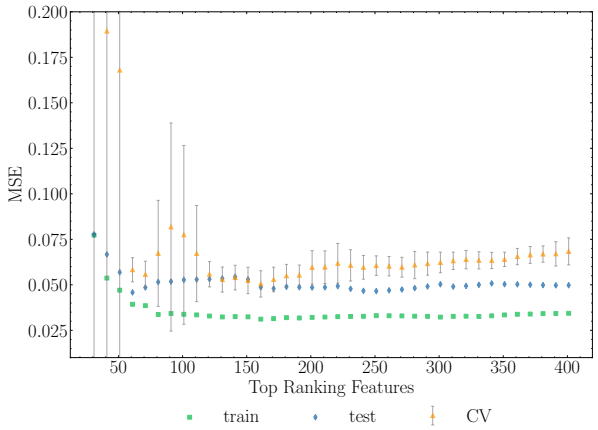


(a) 10 latent variables.

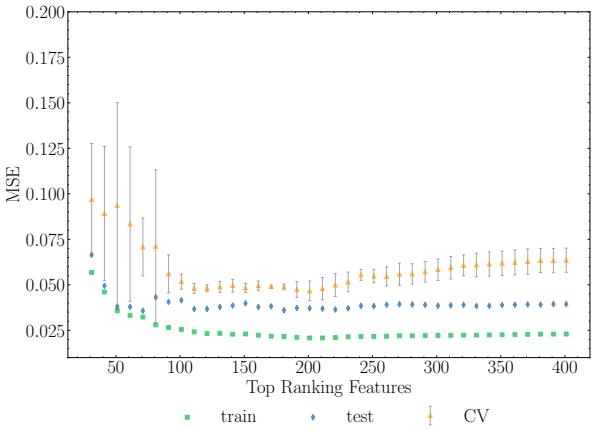


(b) 15 latent variables.

Figure 4.11: PLS with 10 and 15 latent variables over features ranked by PI, Dataset 1.



(a) 20 latent variables.



(b) 25 latent variables.

Figure 4.12: PLS with 20 and 25 latent variables over features ranked by PI, Dataset 1.

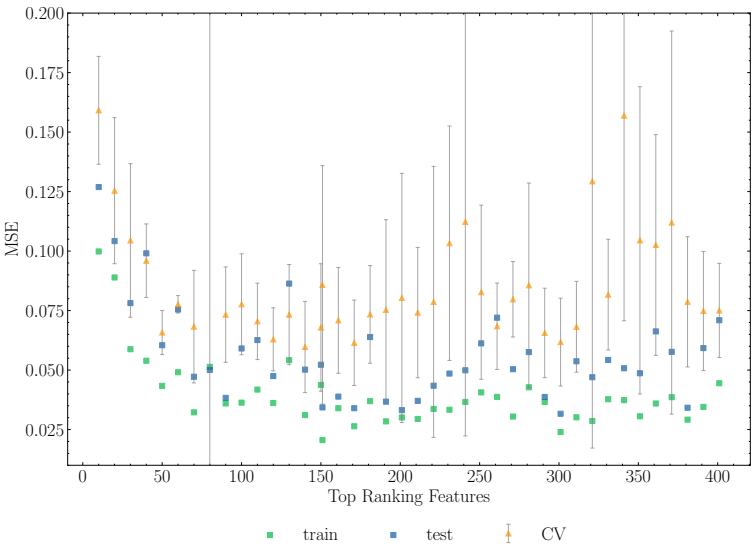


Figure 4.13: ANN MSE over features ranked by PI, Dataset 1.

The results for Dataset 2 produce the same trend; they are presented in Table A.2. It is important to note that PI is slightly less reliable in Dataset 2, particularly with the ANN model. This can be attributed to improper triggering of the stopping criterion, but more precisely due to the fact that Dataset 2 has a significant imbalance problem.

Thus, it is possible to infer that this specific implementation of the algorithm relies on the fact that the important features should be the same among different folds. This is common, but when this is not the case, the algorithm produces unsatisfactory results.

4.1.4.2 Sequential Feature Selection

Despite its conceptual simplicity, SFS presents a significant drawback compared to the other algorithms evaluated in this study: it is inherently greedy. SFS necessitates the evaluation of every possible combination of features to identify the optimal subset, leading to substantial requirements in terms of computational runtime. Specifically, SFS requires considerably more time to execute than any of the other applied algorithms. For instance, in Dataset 1, SFS required approximately 2 hours when running the PLS model and 6 hours for the SVR. This duration is considerably longer than that required by MI, which operates within milliseconds, and mRMR and PI, which operate within minutes. As previously mentioned, the computational complexity of SFS increases quadratically with the number of features. Consequently, applying SFS to Dataset 2 could potentially extend the computational time to several days, and with more complex models such as ANN, the runtime could potentially reach several weeks.

Table 4.4 summarizes the time differences between models. The only unexpected value was that the PI with PLS took longer than with ANN on Dataset 2. This trend was not anticipated and is likely due to the specific implementation or Python packages used, as ANN is a more complex model than PLS and is expected to take longer to run.

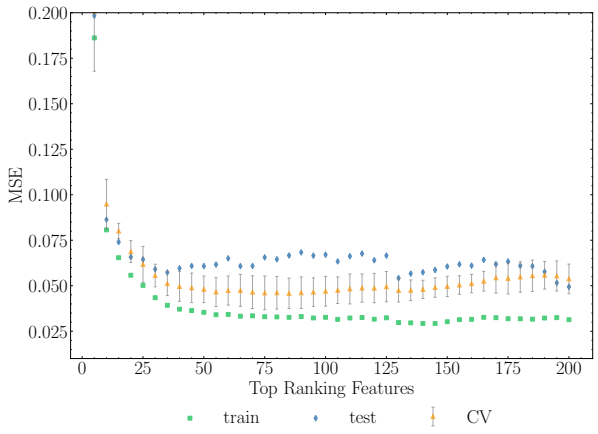


Figure 4.14: PLS with 20 latent variables over features ranked by SFS, Dataset 1.

If SFS consistently showed a decreasing trend with the inclusion of features, its high computational cost would be justified. However, this was not observed in Figure 4.14 nor i other tests, as SFS achieved similar results to other models but took significantly longer.

Table 4.4: Execution times for different feature selection methods on two datasets.

Method	Dataset 1	Dataset 2
mRMR	20 seconds	3 minutes
MI	1 second	3 seconds
PI PLS	1.5 minutes	20 minutes
PI ANN	2 minutes	9 minutes
SFS PLS	2 hours	∼20 hours
SFS SVR	6 hours	-

4.2 Feature Selection Comparison

In the previous sections, the performance of the models when combined with the feature selection algorithms was assessed. The objective of this section is to assess which features were or were not selected by the models with different algorithms and draw conclusions from the observations.

First, an exemplary plot is presented to familiarize the reader with the analysis, serving as a preliminary validation.

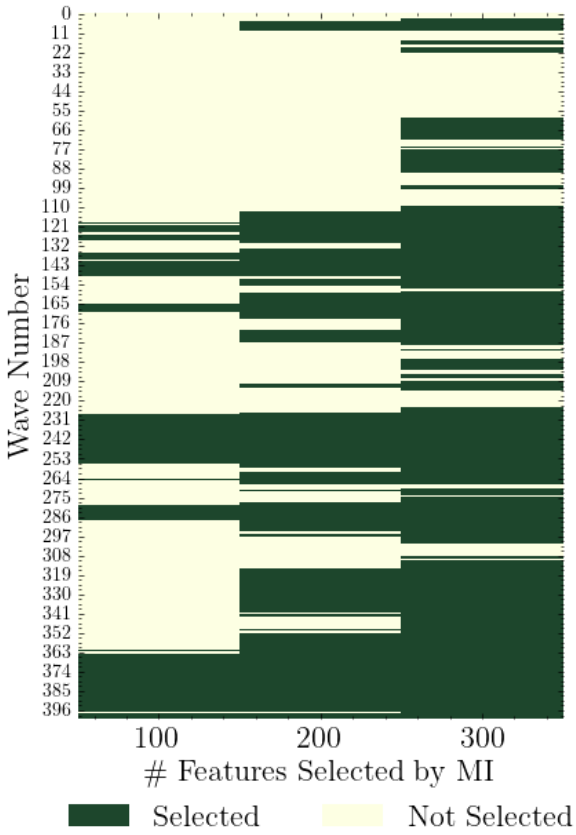
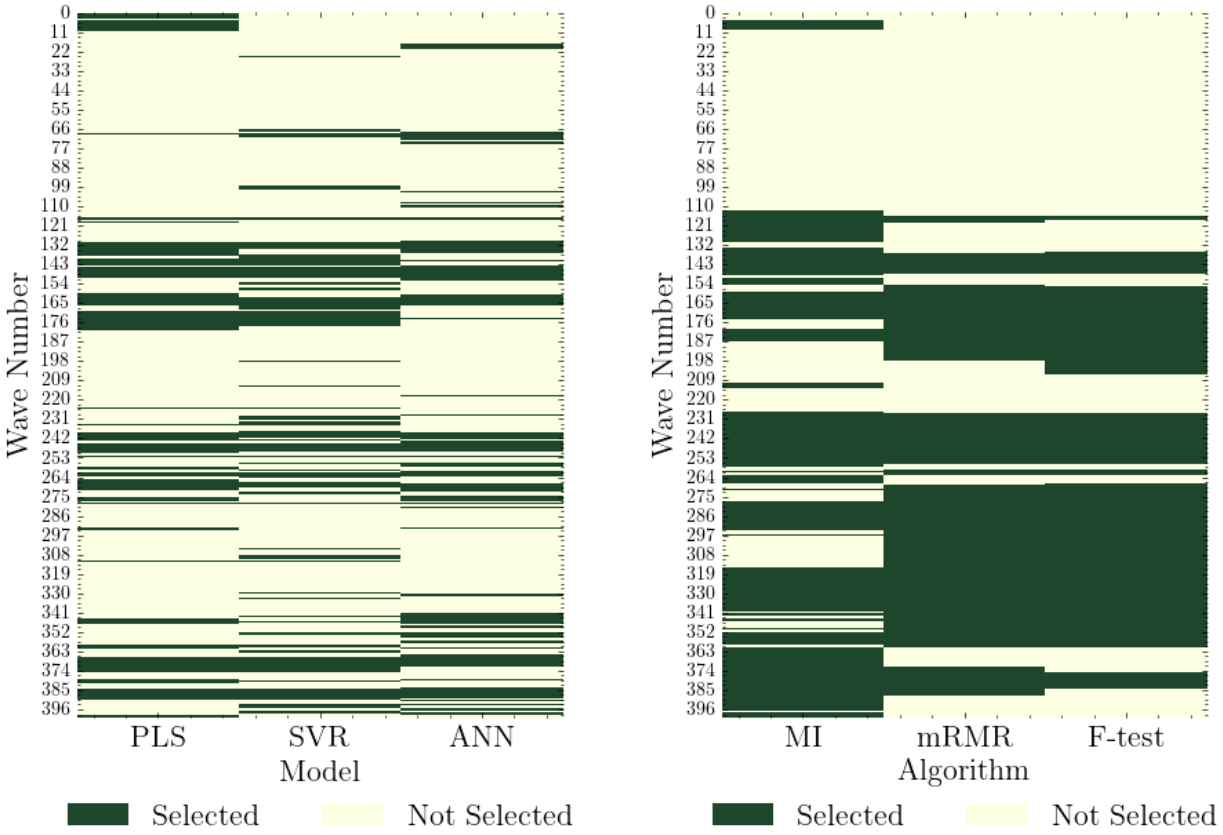


Figure 4.15: Comparison of selected features by different thresholds of MI, Dataset 1.

Figure 4.15 illustrates the features selected by different thresholds of the MI algorithm. The trend indicates that all features selected by the threshold of 100 are also present in the selection with a threshold of 200, and so forth. This confirms that the visualization and selection processes

are functioning as expected. If this were not the case, it would suggest a malfunction in either the visualization, the MI selection, or both.

Figure 4.16a displays the 100 features selected by the PI algorithm for the PLS, ANN, and SVR. The trend is that there are some similarities among the selections, but ultimately, PI selects slightly different features for the different models. This aligns well with expectations because different models perceive different relationships between the data, and PI focuses on the information accessible to the model at hand.



(a) Comparison of 100 selected features by PI with different models, Dataset 1.

(b) Comparison of 200 selected features with different algorithms, Dataset 1.

Figure 4.16: Comparison of selected features, Dataset 1.

Figure 4.16b displays the 200 features selected by mRMR, MI, and an F-Test selection, which is the same as mRMR, but includes only relevance with no regard for redundancy. The first trend, which is very different from the PI algorithm, is that, while there are some similarities, these filter methods seem more prone to selecting neighboring features, unlike PI, which seems to have a slight avoidance of neighboring features. This is likely owing to the nature of Dataset 1, where the more informative features for the target variable are neighbors. This aligns well with the standard notion that neighboring features (or wavenumbers) are correlated, and since a feature is informative to the target, the neighboring features would also be informative. The fact that all these filter methods select similar features is a proof of concept.

An interesting observation is that different selection algorithms achieve similar performances

with the same model. For example, features 130 to 180 achieve the same performance as features 230 to 280. Different subsets of features achieve the same performance using the same model and parameters. This not only implies that the information available in the two distinct feature subsets is the same, but also suggests that there is no single global best subset of features. Since distinct subsets can contain the same information, it is possible that multiple subsets would yield the best performance. This notion also aligns with the fact that there is redundancy present in the spectra.

4.3 Convolutional Neural Networks

In this section, CNN are benchmarked and compared to ANN to assess their viability as an alternative to standard networks. Initially, a single convolution layer was applied immediately after the input layer, followed by the same architecture used in the feature selection assessment. The only difference was the inclusion of this single convolution before the existing architecture. This trial was moderately successful, producing results analogous to the ANN without the convolutional layer.

Following this initial attempt, the Inception module presented in Figure 3.9 was applied. The architecture employed this Inception module after the input layer, followed by a single hidden layer. Implementing this architecture required significant efforts due to the numerous hyperparameters involved. This is particularly challenging because, for both datasets, the model is highly sensitive to small changes in parameters. Many configurations lead to models with high instability, failing to converge even with lower learning rates, high batch sizes, and low dropout rates. This instability was most frequently observed when the resulting feature map had many features, typically when kernel size and stride were small. This ultimately resulted in a very high number of connections between the feature map and the hidden layer, potentially causing instability.

To make the hyperparameter search feasible within a reasonable timeframe, each hyperparameter was individually benchmarked. It was found that most hyperparameters significantly affect performance. However, the following parameters could be fixed to reasonable values as they had a less significant impact on the results: learning rate, dropout rate, number of epochs, L2 regularization coefficient, batch size, and the number of neurons in the hidden layer. The hyperparameters that most significantly affected the results were the output channels, kernel size, and stride of the Inception module. These parameters heavily alter the resulting feature map, leading to varying results.

A grid search was conducted on Datasets 1 and 2. For Dataset 1, the search involved the kernel size of layer 1, the stride of layer 1, and the number of output channels in the convolution from layer 1. For Dataset 2, it involved the number of output channels of the convolution in layer 1 and the convolutions in layer 3, kernel size of layers 1 and 3, and stride of layer 1. It is important to note that this layer notation refers to Figure 3.9, where layer 1 refers to the first convolution, layer 2 to pooling, and so on. The green blocks in the figure have a fixed kernel and stride of size 1. Ultimately, the results obtained were sufficient for a proof of concept. However, it is likely that other combinations of hyperparameters may lead to better performing

models, although at a high cost in computational resources and time.

The initial set of parameters used in the grid search were based on the work by Zhang et al. (2019) [21], whose findings provided a valuable starting points for the hyperparameter tuning process.

Table 4.5: Hyperparameters for the CNN.

	Dataset 1	Dataset 2
Output channels 1	6	4
Output channels 2	4	2
Output channels 3	4	4
Kernel size 1	7	80
Kernel size 2	5	5
Kernel size 3	7	24
Stride 1	3	15
Stride 2	3	3
Stride 3	1	1
Neurons	40	30
Batch size	1000	1000
Dropout rate	0.25	0.1
Reg. coefficient	0.0001	0.0001
Learning rate	0.0001	0.0001
Epochs	2000	2000

Table 4.5 presents the hyperparameters used in the CNN. The notation used refers to the number of output channels from each layer, where output channels 1 relates to layer 1, output channels 2 to layer 2, and so on. Generally, the grid search opted for larger kernel sizes and strides, indicating the redundancy of spectral data. Regarding runtime, CNN were comparable to CNN, likely due to mechanisms such as the leaky ReLU activation function and batch normalization, which speed up the training phase.

Table 4.6: Model performance on Dataset 1 and Dataset 2.

		MSE			R^2		
Dataset	**Model**	**Train**	**Test**	**CV**	**Train**	**Test**	**CV**
1	**ANN**	0.044	0.071	$0.075(20)$	0.955	0.923	$0.923(26)$
	CNN	0.003	0.077	$0.081(11)$	0.997	0.916	$0.917(16)$
2	**ANN**	0.031	0.033	$0.068(37)$	0.705	0.699	$0.354(317)$
	CNN	0.012	0.026	$0.043(11)$	0.890	0.765	$0.604(120)$

From Table 4.6, it is possible to infer that CNN performance is analogous to the ANN in Dataset 1 but slightly superior in Dataset 2, regardless of the imbalance problem. Another important aspect is that the CNN achieve a much better performance on the train set, which could indicate overfitting. However, by analyzing the learning curves (Figure A.3), this is better described as a generalization problem, since there is a consistent gap between the train and test scores. This trend was ever present even with high dropout ratios and regularization coefficients. Specifically,

on the CV score, the CNN was not superior to the PLS. It is possible that the performance could be superior to other models when applied to datasets with highly nonlinear X-Y relationships and numerous features.

Ultimately, this architecture and, by extension, the use of convolutions, present a promising alternative. However, with a significant caveat, which is the difficulty in tuning the parameters. If further studies can simplify the hyperparameter optimization process, CNN could become an even more valuable approach.

5 Conclusions

Feature selection has proven to be a reliable method for enhancing model robustness and reducing runtime, often resulting in simpler models with significantly fewer parameters without compromising performance metrics. However, it generally does not lead to higher determination coefficient or lower Mean Squared Error values. Thus, while feature selection is effective for achieving more robust and simpler models, it does not necessarily improve overall performance. It is more sensible to perform feature selection after achieving satisfactory performance.

All the algorithms evaluated in this study effectively reduced the total number of features with no decrease in performance. The same model achieved similar performances with different subsets of features, suggesting the presence of multiple optimal subsets, aligning with the concept of redundancy. The optimal overall algorithm remains unclear. Among the assessed algorithms, Mutual Information might be the best choice for its ease of implementation and efficiency, while Permutation Importance could be preferable for robustness, provided that the datasets are relatively balanced with respect to the target distribution.

Convolutional Neural Networks show promise and may be a superior option compared to standard Artificial Neural Networks for chemometrics. However, Convolutional Neural Networks require accurate tuning of numerous parameters, which is a significant challenge.

5.1 Limitations

These results may be constrained by the nature of the datasets, which predominantly feature linear X-Y relationships, typical of Near-Infrared spectroscopy. Raman spectroscopy, which often involves nonlinear relationships, could potentially enhance the applicability of some feature selection algorithms or CNN. Additionally, other promising feature selection algorithms have yet to be evaluated.

5.2 Future Work

Exploring other hyperparameter tuning strategies, such as Bayesian hyperparameter optimization, could address the tuning difficulties associated with Convolutional Neural Networks, despite their high computational demands and complexity. The architecture used in this study is promising, but other alternatives could also be successful. Further investigation of other promising feature selection algorithms, such as genetic algorithms, is warranted.

Many articles suggest that Convolutional Neural Networks can process raw data without specific preprocessing or normalization. Although not assessed in this work, understanding the reliability of this approach in different situations would be valuable.

Additionally, studying the correlation between features and targets from a chemical perspective, focusing on the chemical bonds associated with specific wave numbers, could be valuable for developing heuristics to tackle specific prediction problems. This study could streamline the prediction process by identifying which chemical bonds and wave numbers are relevant for predicting certain variables, thereby eliminating the need for a feature selection step.

6 Assessment of The Work Done

This chapter evaluates the objectives achieved, contributions to sustainable development goals, and overall outcomes of the project.

6.1 Objectives Achieved

This work successfully evaluated feature selection techniques, enhancing the understanding of their capabilities. Additionally, Convolutional Neural Networks were successfully assessed, showing promise but indicating that further research could significantly improve their viability.

6.2 Contribution to the Sustainable Development Goals

In today's industrial landscape, advanced data analysis techniques are essential for sustainable development. This work explores the use of chemometrics and convolutional neural networks to improve industrial processes. Table 6.1 outlines how these techniques may contribute to several Sustainable Development Goals (SDGs) by enhancing industrial quality control, innovation, resource optimization, and energy efficiency. These incremental advancements pave the way for future innovations in industrial sustainability.

Table 6.1: Contributions to SDGs.

SDG	Target	Contribution	Performance Indicators
3	3.8	Improvement in industrial quality control and safety through chemometric techniques.	Number of improved processes, reduction in product recalls.
9	9.5	Advancement of chemometric techniques for spectral data analysis, supporting industrial innovation and infrastructure improvement.	Number of industries adopting new techniques, enhancement in data analysis efficiency.
12	12.2	Optimization of resources in chemical and pharmaceutical processes through feature selection methods, promoting sustainable production.	Reduction in resource usage, increase in production efficiency.
13	13.3	Reduction in computational resources and energy consumption in continuous data analysis processes, contributing to lower carbon footprints in industrial activities.	Decrease in energy consumption, reduction in carbon emissions.

6.3 Final Assessment

Data analytics is a highly engaging yet broad field with numerous promising possibilities to explore. Due to the many available options, identifying the best approach and planning accordingly

can be challenging. Another significant challenge lies in the selection of data sources; despite the availability of many datasets, the chosen one had a significant imbalance problem, requiring preventive strategies. This work demanded considerable effort due to its challenging nature and would not have been possible without the commendable guidance from the advisors.

References

[1] Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., Hamprecht, F. A., "A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data," *BMC Bioinformatics*, vol. 10, p. 213, 2009.

[2] Liu, H., Motoda, H., *Feature Selection for Knowledge Discovery and Data Mining*. Springer, 1998.

[3] Yang, J., Xu, J., Zhang, X., Wu, C., Lin, T., Ying, Y., "Deep learning for vibrational spectral analysis: Recent progress and a practical guide," *Analytica Chimica Acta*, vol. 1081, pp. 6–17, 2019.

[4] Balabin, R. M., Smirnov, S. V., "Variable selection in near-infrared spectroscopy: Benchmarking of feature selection methods on biodiesel data," *Analytica Chimica Acta*, vol. 692, pp. 63–72, 2011.

[5] Teófilo, R. F., Martins, J. P. A., Ferreira, M. M. C., "Sorting variables by using informative vectors as a strategy for feature selection in multivariate regression," *Journal of Chemometrics*, vol. 23, pp. 32–48, 2009.

[6] Xu, S., Zhao, Y., Wang, M., Shi, X., "Determination of rice root density from vis–nir spectroscopy by support vector machine regression and spectral variable selection techniques," *CATENA*, vol. 157, pp. 12–23, 2017.

[7] Chen, H., Tan, C., Lin, Z., Wu, T., "Classification and quantitation of milk powder by near-infrared spectroscopy and mutual information-based variable selection and partial least squares," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 189, pp. 183–189, 2018.

[8] Knief, P., Clarke, C., Herzog, E., Davoren, M., Lyng, F. M., Meade, A. D., Byrne, H. J., "Raman spectroscopy - a potential platform for the rapid measurement of carbon nanotube-induced cytotoxicity," *Analyst*, vol. 134, pp. 1182–1191, 2009.

[9] Zhao, J., Zeng, H., Kalia, S., Lui, H., "Wavenumber selection based analysis in raman spectroscopy improves skin cancer diagnostic specificity," *Analyst*, vol. 141, pp. 1034–1043, 2016.

[10] Xie, C., Xu, N., Shao, Y., He, Y., "Using ft-nir spectroscopy technique to determine arginine content in fermented cordyceps sinensis mycelium," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 149, pp. 971–977, 2015.

[11] Schwanninger, M., Rodrigues, J. C., Gierlinger, N., Hinterstoisser, B., "Determination of lignin content in norway spruce wood by fourier transformed near infrared spectroscopy and partial least squares regression. part 1: Wavenumber selection and evaluation of the selected range," *Journal of Near Infrared Spectroscopy*, vol. 19, pp. 319–329, 2011.

[12] Xu, S., Lu, B., Baldea, M., Edgar, T. F., Nixon, M., "An improved variable selection method for support vector regression in nir spectral modeling," *Journal of Process Control*, vol. 67, pp. 83–93, 2018.

[13] Kaneko, H., Kono, S., Nojima, A., Kambayashi, T., "Transfer learning and wavelength selection method in nir spectroscopy to predict glucose and lactate concentrations in culture media using vip-boruta," *Analytical Science Advances*, vol. 2, pp. 470–479, 10 2021.

[14] Ma, X.-H., Chen, Z.-G., Liu, J.-M., "Wavelength selection method for near-infrared spectroscopy based on max-relevance min-redundancy," *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, vol. 310, p. 123933, 2024.

[15] Cai, Y., Yang, C., Xu, D., Gui, W., "Quantitative analysis of stibnite content in raw ore by raman spectroscopy and chemometric tools," *Journal of Raman Spectroscopy*, vol. 50, pp. 454–464, 2019.

[16] Lanza, I., Conficoni, D., Balzan, S., Cullere, M., Fasolato, L., Serva, L., Contiero, B., Trocino, A., Marchesini, G., Xiccato, G., Novelli, E., Segato, S., "Assessment of chicken breast shelf life based on bench-top and portable near-infrared spectroscopy tools coupled with chemometrics," *Food Quality and Safety*, vol. 5, 2021.

[17] Shizhuang, W., Sheng, C., Miao, L., Xinhua, Z., Shouguo, Z., Jian, Z., Jin, C., Lei, C., "Quantitative analysis of thiram based on sers and plsr combined with wavenumber selection," *Analytical Methods*, vol. 6, pp. 242–247, 2014.

[18] Qiu, Y., Kuang, C., Liu, X., Tang, L., "Single-molecule surface-enhanced raman spectroscopy," *Sensors*, vol. 22, 2022.

[19] Acquarelli, J., van Laarhoven, T., Gerretzen, J., Tran, T. N., Buydens, L. M., Marchiori, E., "Convolutional neural networks for vibrational spectroscopic data analysis," *Analytica Chimica Acta*, vol. 954, 2017.

[20] Liu, J., Osadchy, M., Ashton, L., Foster, M., Solomon, C. J., Gibson, S. J., "Deep convolutional neural networks for raman spectrum recognition: A unified solution," *Analyst*, vol. 142, pp. 4067–4074, 11 2017.

[21] Zhang, X., Lin, T., Xu, J., Luo, X., Ying, Y., "Deepspectra: An end-to-end deep learning approach for quantitative spectral analysis," *Analytica Chimica Acta*, vol. 1058, 2019.

[22] Cui, C., Fearn, T., "Modern practical convolutional neural networks for multivariate regression: Applications to nir calibration," *Chemometrics and Intelligent Laboratory Systems*, vol. 182, 2018.

[23] Fan, J., Han, F., Liu, H., "Challenges of Big Data analysis," *National Science Review*, vol. 1, pp. 293–314, 2014.

[24] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] Ding, C., Peng, H., "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, pp. 185–205, Apr 2005.

[26] Zhao, Z., Anand, R., Wang, M., "Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2019, pp. 442–452.

[27] Cover, T. M., Thomas, J. A., *Elements of Information Theory*. John Wiley & Sons, Ltd, 2005.

[28] Kraskov, A., Stögbauer, H., Grassberger, P., "Estimating mutual information," *Phys. Rev. E*, vol. 69, p. 066138, 2004.

[29] Ross, B. C., "Mutual information between discrete and continuous data sets," *PLOS ONE*, vol. 9, pp. 1–5, 2014.

[30] Kozachenko, L. F., Leonenko, N. N., "Sample estimate of the entropy of a random vector," *Probl. Peredachi Inf.*, vol. 23, pp. 9–16, 1987.

[31] Breiman, L., "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[32] Wold, S., Sjöström, M., Eriksson, L., "Pls-regression: a basic tool of chemometrics," *Chemometrics and Intelligent Laboratory Systems*, vol. 58, pp. 109–130, 2001.

[33] Elsken, T., Metzen, J. H., Hutter, F., "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, pp. 1–21, 2019.

[34] Cristianini, N., Shawe-Taylor, J., *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[35] Goodfellow, I., Bengio, Y., Courville, A., *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[36] Sanderson, G., "Videos repository for 3blue1brown," https://github.com/3b1b/videos, accessed August 28, 2024.

[37] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[38] Yan, H., Xu, Y.-C., Siesler, H. W., Han, B.-X., Zhang, G.-Z., "Hand-held near-infrared spectroscopy for authentication of fengdous and quantitative analysis of mulberry fruits," *Frontiers in Plant Science*, vol. 10, p. 1548, 2019.

[39] Nair, V., Hinton, G. E., “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, ser. ICML’10. Omnipress, 2010, pp. 807–814.

[40] Kingma, D. P., Ba, J. L., “Adam: A method for stochastic optimization,” 2015.

A Appendix: Complementary Figures and Tables

Table A.1: Worse score, generally on iterations with lower epochs.

Index	Test MSE	Epochs	Index	Test MSE	Epochs
401	0.029	1200	221	0.035	2103
391	0.043	763	211	0.017	1431
381	0.046	1092	201	0.024	2103
371	0.034	1617	191	0.038	1431
361	0.024	1320	181	0.029	1691
351	0.087	1313	171	0.041	1610
341	0.040	1094	161	0.096	491
331	0.046	1398	151	0.054	911
321	0.047	2094	141	0.035	1369
311	0.026	719	131	0.050	1256
301	0.075	498	121	0.039	2006
291	0.071	1199	111	0.051	820
281	0.029	1652	101	0.038	1629
271	0.052	967	91	0.031	723
261	0.027	1477	81	0.036	1713
251	0.064	987	71	0.030	1319
241	0.036	1198	61	0.076	1753
231	0.045	1829	51	0.073	1357

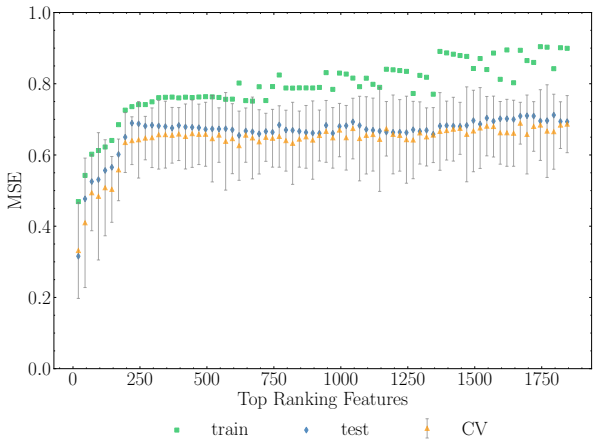


Figure A.1: PLS MSE with the best number of latent variables over features ranked by MI, Dataset 2.

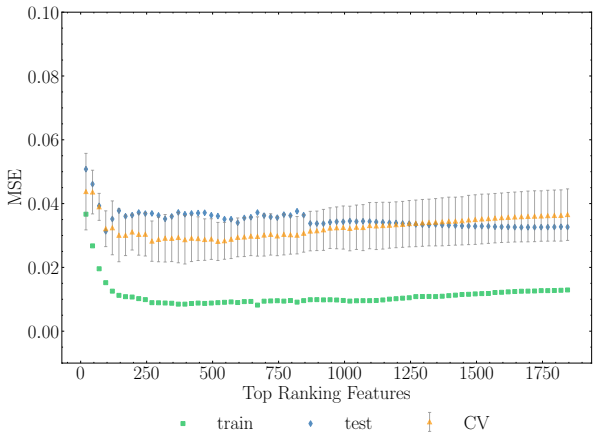


Figure A.2: PLS R^2 with 16 latent variables over features ranked by PI, Dataset 2.

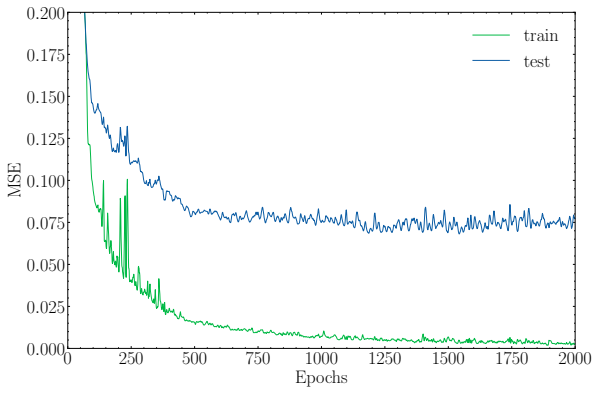


Figure A.3: CNN generalization problem

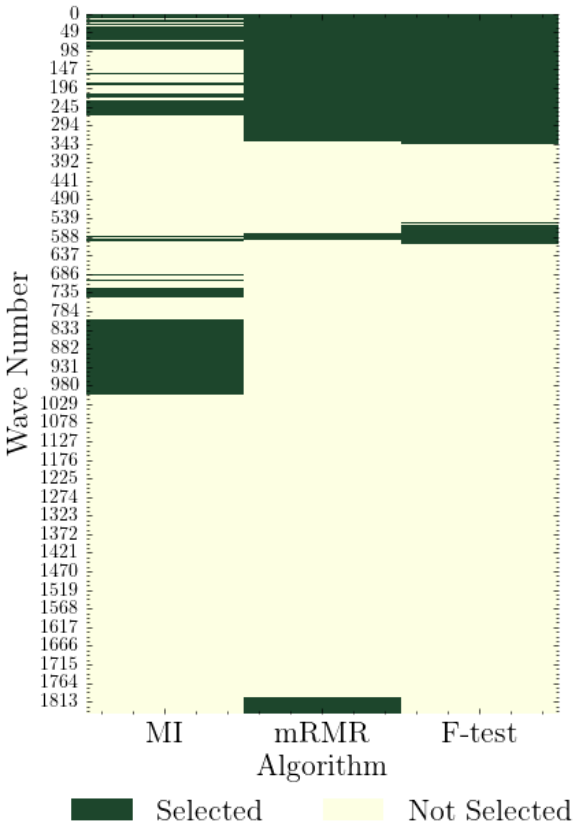


Figure A.4: Comparison of 500 selected features by different algorithms, Dataset 2.