THE UNIVERSITY OF LIVERPOOL

PHD THESIS

The Development of Machine Learning Methods for Prognostic Tools in Head and Neck Cancer

---

Author:

Conor Whitley

Supervisors:

Dr. David Martin

Dr. Steve Barrett

Professor Marta Garcia-Finana

Dr Ruwanthi Kolamunnage-Dona

Abstract

Contents

List of Figures

List of Tables

1 FTIR Preprocessing Pipeline Optimisation

1.1 Introduction

In recent decades, vibrational spectroscopy has emerged as a useful analysis method applicable to a range of specimens, from food and pharmaceutical samples to security and medical applications. Vibrational spectroscopy techniques interrogate the molecular structure of a specimen by utilising the infrared radiation, either directly (infrared absorption spectroscopy) or indirectly (Raman scattering spectroscopy). The absorption of discrete photon energies by molecular vibrational modes within the sample gives rise to characteristic spectral profiles, which can be associated with the inherent chemistry of the sample under interrogation.

1.1.1 Optimisation

Despite the undeniable promise of the field, it is somewhat hampered by the lack of consensus in precisely how to preprocess the data preprocessing subsequently analyse it. The number of available preprocessing methods and predictive models is truly vast, each method typically has one or more parameters associated with it exacerbating the problem even further.

Analysis of spectroscopic data is typically a multi-step procedure, starting with preprocessing, and ultimately the employment of pattern recognition and machine learning tools to classify the data into distinct groups. Preprocessing is a vital step in the analysis process of IR data, as it has been shown to generally increase performance of classification models [1] allowing them to generalise more effectively to larger clinical cohorts, and to increase the interpretability of results.

FTIR and Raman data-sets are highly dimensional. In the case of Raman mapping and FTIR imaging, sample numbers are very large, with data-sets comprising large numbers of patients typically having tens of thousands of spectra. A further motivating factor known as the 'no free lunch' (NFL) theorem states that 'any two optimization algorithms are equivalent when their performance is averaged across all possible problems' [2]. This was stated generally for optimisation problems; due to the close relationship between optimisation and machine learning methods the same theorem applies [3]. The implications of the NFL theorem would require a thorough search of multiple machine learning models to ensure some degree of certainty that the task is possible.

An optimisation protocol that can efficiently search across a highly dimensional space would be of considerable benefit to users of multivariate analysis techniques. One approach [4] utilised a genetic algorithm (GA) to perform an optimisation search procedure to determine more effective processing pipelines. A single 'individual' in this framework was represented as a processing pipeline; many generations of preprocessing pipelines were allowed to 'evolve' in a manner analogous to Darwinian evolution. The 'fittest' individuals which influence subsequent generations were chosen to be those with the lowest prediction error, hence directing the process towards optimal values. The optimised pipeline resulted in 16% reduction in the model error compared with the raw data model. Similar work [5] utilised a GA approach to assist with feature selection. The authors were interested in using Curie-point pyrolisis mass spectrometry to differentiate between bacterial spore samples. The highly dimensional nature of these datasets makes interpretation prohibitively difficult; the GA approach was used to select a subset of these features for further analysis. A trial-and-error approach was more recently reported [6] which trialed every permutation of preprocessing steps within a defined search space on an ATR-FTIR biofluid dataset comprises patients with varying types of brain cancer. The authors utilised random forest (RF) support vector machines (SVM) classifiers, however the hyperparameters of the final classifiers were not optimised in this approach. A brute force approach will cover every possible combination but the number of combinations grows extremely quickly and is generally infeasible as a strategy.

This work proposes a novel approach to objectively optimise an effective preprocessing and classification pipeline. We perform a Bayesian hyperparameter search on a number of candidate pipelines using a parallel computing approach — more efficiently searching all possible solutions. The focus of this paper will be on FTIR imaging data; but the approach is in theory generalisable to *any* pipeline type inference problem.

1.2 Theoretical

1.2.1 Preprocessing

Preprocessing of FTIR data can be broken into a number of discrete steps, with each step designated to mitigate unwanted spectral aberrations and measurement artefacts. Using modern object-oriented programming languages, preprocessing step can be encapsulated as a transformer object — an abstract representation of a preprocessing step. A transformer will typically take a dataset as input, perform the transformation associated with it, and then output the transformed data. A sequence of transformers with or without a final estimator can be visualised as a pipeline. A pipeline consists of a sequence of transformers which take data as an input and pass the data through each transformer sequentially until a final result is obtained. Such a preprocessing sequence is briefly set out below.

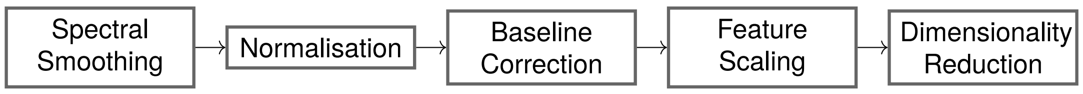

FIGURE 1.1: Preprocessing pipeline flowchart

Spectral smoothing - *Accounts for high frequency noise in the data*. This unwanted noise may have instrumental, environmental, or sample origins. There are several associated methods, including the commonly used Savitzy-Golay [7], whereby a polynomial is fit to a local moving window of specified length. Other methods such as PCA de-noising and FFT filtering are also commonly applied

Normalisation - *Accounts for unimportant changes in absolute absorbances*. This important step accounts for absolute differences in absorption according to the Beer-Lambert law [8] — which is to be expected with a biologically diverse data-set from multiple samples. This step helps to mitigate the effect of sample thickness on the analysis — a potential confounding factor.

Baseline correction - *Accounts for variations* . Spectra are often superimposed on a non-linear baseline as a result of background scattering interference effects. Methods such as rubberband correction and spectral differentiation are often applied to compensate for these effects.

Feature scaling - *Ensures variables all lie within the same range*. A scaling step is sometimes applied in order to scale the absorbance for each wavenumber to a common range. This can often increase the performance of some classifiers and is sometimes a required preprocessing step for dimensionality reduction algorithms.

Dimensionality reduction - *A reduction in the number of predictive variables*. The reduction of the dimensionality of the data to a much lower space often facilitates more robust classifier results and can mitigate issues arising from correlated variables. Feature selection/extraction techniques such as PCA [9] and forward feature selection (FFS) [10] are often used to reduce the number of variables while retaining maximum important information.

In addition to numerous preprocessing methods, each may have intrinsic parameters, referred to from here on as *hyperparameters*, which must also be determined. Exhaustively trialling every possible combination of preprocessing transformations to a dataset is often prohibitive. Whilst this would yield the true optimal combination of methods, it would be at a high computational cost.

Whilst it is possible to exhaustively trial all preprocessing transformations to find the true optimal combination of methods, this may come at prohibitive computational cost. For instance, consider a case where there are five preprocessing steps, each containing five methods to select from. Each of these methods has a hyperparameter space that spans twenty possible values. The total number of pipelines to construct and evaluate and is equal to 10^{10}, rendering this brute-force approach prohibitive when faced with a large, multidimensional search space.

There is demand for an optimisation protocol that can efficiently and intelligently search across a high dimensional space. One approach [4] utilised a genetic algorithm (GA) by which generations of preprocessing sequences were allowed to 'evolve' in a manner analagous to Darwinian evolution. The 'fittest' pipelines were allowed to cross-over with each other to produce offspring which can mutate and cross-over to produce more generations. The optimised pipeline resulted in 16% reduction in the model error compared with the raw data model. GA does not scale well with complexity as each generation is dependant on the previous, therefore improvements in runtime by means of parallelisation is not possible. A trial-and-error approach was more recently reported [6] which trialed every permutation of preprocessing steps within a defined search space on an ATR-FTIR biofluid dataset of brain cancer patients, prior to classification using either random forest (RF) or RF/GA fed support vector machines (SVM). This brute force approach is certainly very thorough, but can also lead to prohibitive runtimes. Furthermore, the hyperparameters of the final estimators were not optimised in this approach.

This work proposes a novel approach to objectively optimise a preprocessing and classification pipeline. We combine parallel processing with Bayesian hyperparameter search to efficiently search a large parameter space. To test the framework on an existing problem an FTIR-imaging dataset comprising a number of patients and over 100,000 spectra is used.

1.2.2 Bayesian hyperparameter Search

In order to efficiently search for optimal pipeline configurations, it was necessary to perform a Bayesian hyperparameter search due to the computational expense of evaluating each pipeline. An open-source python library *scikit-optimize* [11] was used to leverage a Gaussian process (GP) regression over the parameter space associated with each job.

A GP is an efficient method of searching for a maxima or minima over a complicated function. The GP model is utilised here to approximate the loss function with limited data. The loss function is the score obtained when a proposed set of hyperparameters is used. The search for an optimal set of hyperparameters is summarised in Equation (1.1).

$$\boldsymbol{\theta}^{*} = \operatorname*{argmin}_{\theta} f(\boldsymbol{\theta}) \tag{1.1}$$

Where $\boldsymbol{\theta}^{*}$ is the optimal hyperparameter configuration, and $f(\boldsymbol{\theta})$ corresponds to the process of training and evaluation of the pipeline in question using the hyperparameter vector $\boldsymbol{\theta}$. This framework is well-suited to the problem at hand which can be described as a sequential model-based optimisation task. The loss function is estimated sequentially; using previous evaluations to determine optimal hyperparameter configuration to evaluate next. In order to evaluate the next point which will result in the greatest improvement in the loss function given all previous evaluations and current estimates of the space; the expected improvement criterion is used [12]:

$$EI(\boldsymbol{\theta}) = (\mu(\boldsymbol{\theta}) - f(\widehat{\boldsymbol{\theta}}))\Phi(Z) + \sigma(\boldsymbol{\theta})\phi(Z) \tag{1.2}$$

$$Z = \frac{\mu(\boldsymbol{\theta}) - f(\widehat{\boldsymbol{\theta}})}{\sigma(\boldsymbol{\theta})} \tag{1.3}$$

Where $\Phi(z)$, and $\phi(z)$, are the cumulative density function and probability density function of a multivariate normal distribution respectively.

1.2.3 Gaussian Processes

A GP regression seeks to estimate a distribution over an infinite set of candidate functions over a noisy loss function [12]. A GP frames the problem in such a way that each point in the optimisation space is considered to be a dimension in a multivariate Gaussian, described by a mean function eq. (1.4) and covariance matrix eq. (1.5):

$$m(x) = \mathbb{E}[f(x)] \tag{1.4}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{1.5}$$

Where a GP is defined as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{1.6}$$

The mean function is the approximation of the underlying hyperspace which we wish to estimate. Whereas the covariance function quantifies the relationship between the points in this space. The covariance function $k(x, x')$ can be represented by a number of different functions and can be used to instill prior knowledge of the relationship between data points. A commonly used covariance function is the *Matern* kernel:

$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^{v} K_{\nu} \left(\frac{\sqrt{2\nu} r}{l} \right) \tag{1.7}$$

Where

$$r = x - x' \tag{1.8}$$

Where K_ν is a modified Bessel function of order ν and l is the parameter governing the length of the relation between data points. The Matern kernel is able to cope with a noisy loss function and is the default kernel for the GP optimisation in skopt's BayesSearchCV. A GP is generally considered to be a non-parametric method, however it is conceptually helpful to consider a GP as having an *infinite* number of parameters. This is due to the fact that a GP instead seeks to estimate the posterior distribution over an *infinite* number of potential functions. This can be contrasted with a typical random variable as instead of drawing a scalar value from the corresponding distribution, a *function* is drawn instead. The function is drawn from a Gaussian distribution of mean eq. (1.4) and covariance eq. (1.5).

Toy example

In order to demonstrate the sequential optimisation procedure described above, and re-enforce the intuition behind a GP; a *toy* example has been constructed, a visual representation is shown in fig. 1.2. Shown in plot (a) is the true underlying function that is to be approximated. Plots (b) and (c) show the mean and standard deviation at each point respectively. Plot (d) shows the expected improvement at each point in the space given by eq. (1.2).

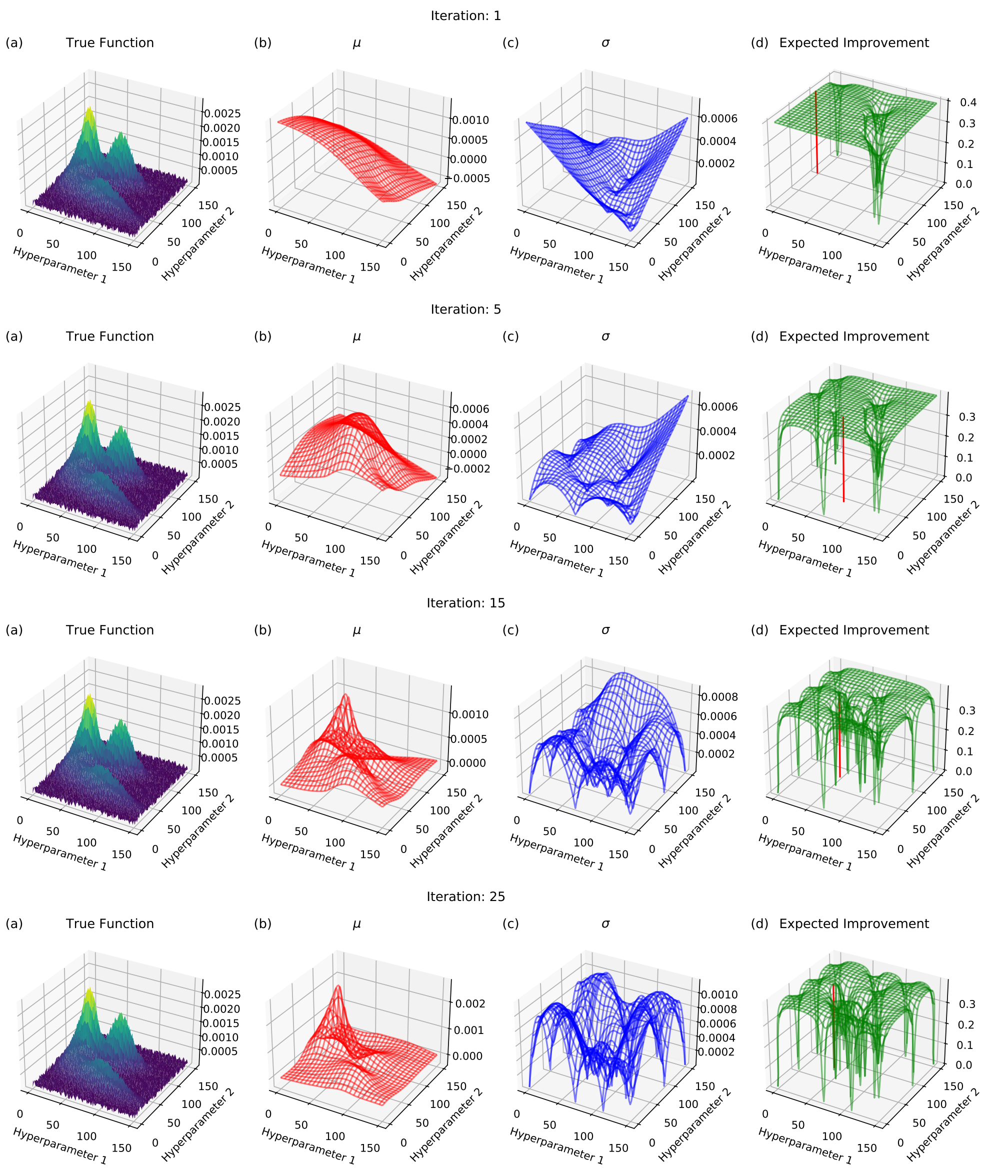

FIGURE 1.2: Bayesian hyperparameter search using a GP

The optimiser is initialised with 5 random points to assist with convergence of the algorithm. These initial points are then used to fit a GP regression at the first step. The GP regression (fig. 1.3) then predicts the mean function (b) and standard deviation (c) at each point in the search space given. The expected improvement is then calculated using the observed data at that step. The maximum expected improvement is then taken as the point which will next be sampled. This next point is then taken with all previous data and the process is repeated for the desired number of iterations. The hyperparameter configuration chosen is then taken to be that which resulted in the most optimal score from the sequence of previous evaluations.

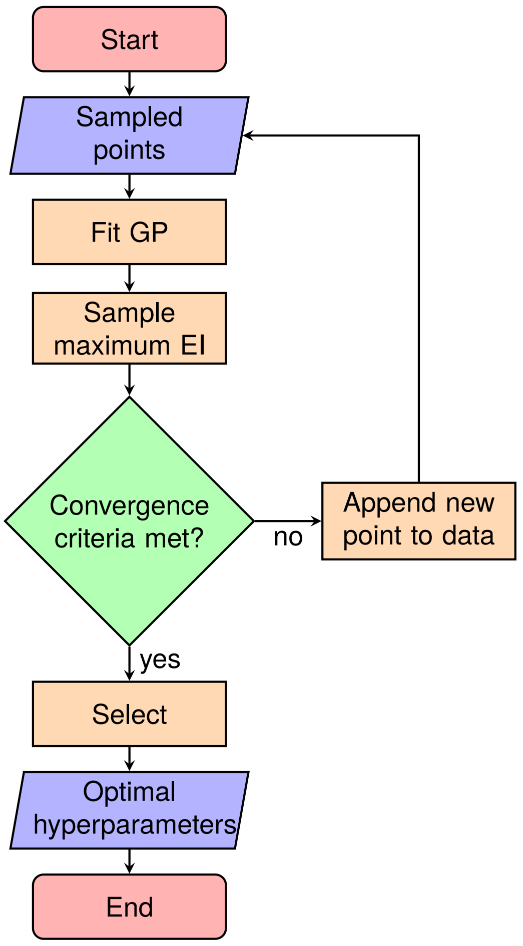

FIGURE 1.3: GP hyperparameter optimisation flowchart showing the overall process.

The number of iterations can be dictated by a number of criteria such as: the convergence of the loss score, a preferred number of iterations, or a set amount of time. The convergence criteria is generally subject to the constraints of the computing resources available, but if resources are plentiful a convergence criteria is usually used.

Figure 1.4 shows a comparison between the mean function represented in fig. 1.2 (a), and the true function we wish to approximate (b).

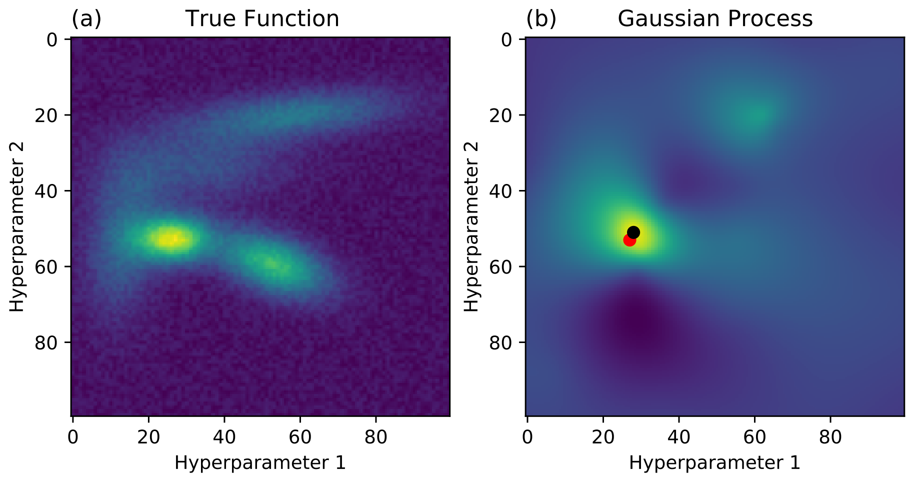

FIGURE 1.4: A comparison of true function to GP regression approximation. The optima predicted by the GP regression (red dot) is very close to that of the true function (black dot).

Shown in red and black dots are the global optimum of the true function, and maximum of the GP mean function. Whilst a GP is able to approximate a high-dimensional loss surface well, it only serves to *direct* the search process. The optimal hyperparameter set chosen is selected as the set which was *known* to yield a low loss score through evaluation. This avoids the use of the somewhat speculative maximum given by the GP mean function, which can be particularly important when considering very highly dimensional search spaces where the number of points sampled is relatively low compared to the entire volume of the space.

1.3 Methods

To utilise the framework, first the methods and hyperparameter search spaces they wish to trial must be input. Also defined in this stage is the order in which the steps are applied, with an estimator always occupying the final step. At this point, it is instructive to define the validation procedure to use in the scoring stage. Protocols such as k-fold cross validation and leave-one-out cross validation are preferential over a single train-test split to mitigate the risk of overfitting. A completely independent set of data is put aside to test the final, optimised model. Each possible combination of methods is generated and distributed to a network of computers. The initial release of the framework is optimised for those utilizing the well established HTCondor service [13]. HTCondor is an open source high throughput package which enables the user to distribute parallelizable computationally expensive tasks (jobs) to a pool of idle computers on a local network, a method known as 'cycle-scavenging'. A flowchart summary of the process is shown in fig. 1.5.

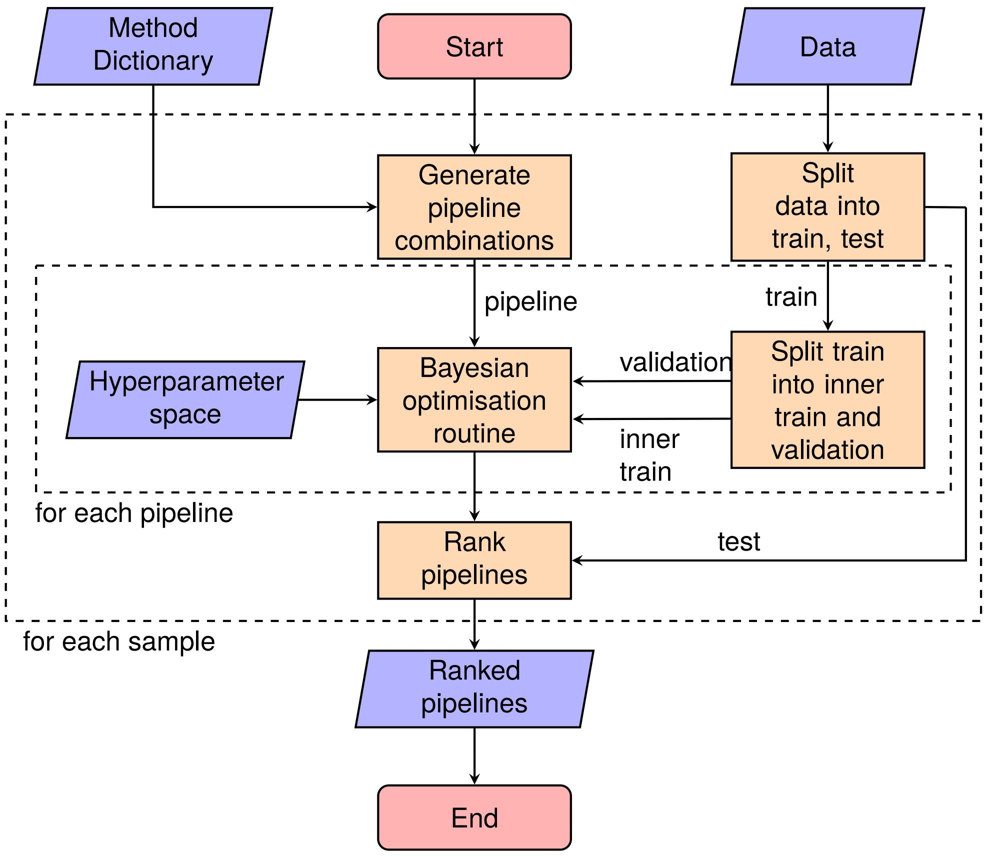

FIGURE 1.5: Flowchart of overall optimisation process

The number of jobs is directly related to the number of methods n belonging to each step i by the following:

$$n_{pipelines} = \prod_{i=0}^{n_{parameters}} n_i \tag{1.9}$$

Each distributed job is a unique combination of preprocessing transformers and final estimator. Contained within the job configuration is the set of search spaces associated with the hyperparameters, which can be initialised as one of three different types:

- **Categorical** — hyperparameter values can be any from an unordered list.
- **Continuous space** — hyperparameters are drawn from a defined probability distribution.
- **Integer space** — hyperparameters are discretised values from a given range.

Different sampling distributions can be specified when viable hyperparameter values span a wide range. Specifying sampling distributions for hyperparameters allows the user to provide prior knowledge of feasible values; or avoided entirely by specifying a uniform distribution.

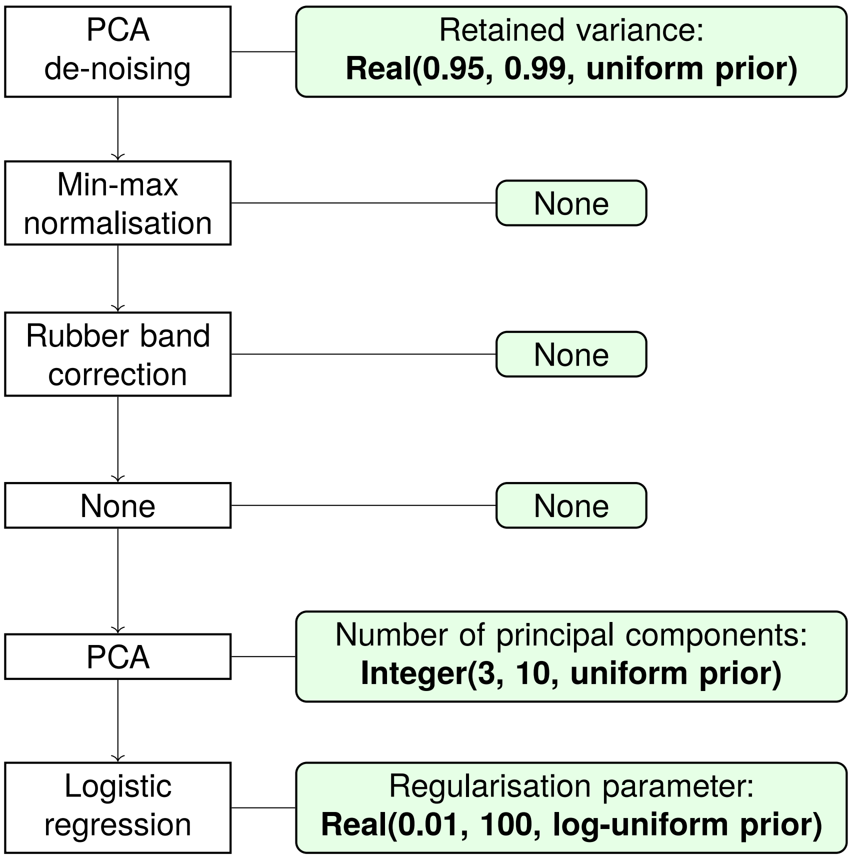
PCA de-noising
Retained variance: **Real(0.95, 0.99, uniform prior)**
Min-max normalisation
None
Rubber band correction
None
None
None
PCA
Number of principal components: **Integer(3, 10, uniform prior)**
Logistic regression
Regularisation parameter: **Real(0.01, 100, log-uniform prior)**

FIGURE 1.6: An example pipeline showing each step with associated hyperparameter search arguments (green).

As an example, consider a processing pipeline consisting of PCA-de-noising, followed by min-max normalisation, PCA, and ending with a logistic regression classifier. Three steps have hyperparameters which require tuning: the explained variance of the PCA de-noising method, the number of principal components retained by the PCA transform, and the regularisation parameter in logistic regression. A reasonable range of values to optimise over for the explained variance would be 0.95-0.99 (95-99%) to eliminate low variance noise; the space should be initialised as **Real(0.95, 0.99, uniform prior)**. The regularisation parameter value of the logistic regression classifier is inversely proportional to the regularisation strength — smaller values correspond to a stronger regularisation effect and mitigate the potential for overfitting. Optimal values exist on a much wider domain, nominally taking up values between 10^{-2} and

10^2, therefore the search space argument is initialised as **Real(0.01, 100, log-uniform prior)** sampling from a log-uniform distribution.

The initialised job now begins the Bayesian search regime, through which the hyperparameter space is sequentially sampled, updating the loss function at each iteration and informing the subsequent search choices taken as the maximum expected improvement in the loss score eq. (1.2). The number of iterations, loss function, and validation protocol, are all pre-defined parameters which can be selected based on the size of the search space and type of optimisation problem. In the case of a classification task with a large parameter space, a large number of iterations will likely be needed. Once a set of hyperparameters has been found the fine tuned pipeline is validated on an unseen dataset to prevent information leakage and thus overly optimistic results. The completed jobs are then aggregated and ranked according to the mean validation AUC score.

1.4 Results

In order to test the framework an evaluation was performed on the same FTIR dataset described in **??**. The objective was to obtain the optimised pipeline with the best mean performance across a number of train-test splits. The task was to predict the whether a patient would live beyond, or less than one year of the most recent review date.

Using a test set of preprocessing methods 576 unique pipelines were evaluated using the optimisation routine. Approximately one third of patients were held out for final model evaluation, with the remaining patients used for model training and optimisation. The loss function was the aggregated mean AUC score of three folds patient stratified folds; the optimiser iteration limit was set to 50.

The jobs were distributed to the HTCondor framework at the University of Liverpool, UK. Each of the 1900 PCs on the network is equipped with an Intel Core i3 (quad-core) processor running at 3.3 GHz, 8 GB RAM and 120 GB storage. Completed jobs were extracted from HTCondor and the results for each permutation across the 50 train-test splits were aggregated in order to compare average results. Pipelines were ranked according to the mean AUC score across the 50 iterations; classification scores for these pipelines are shown in fig. 1.7.

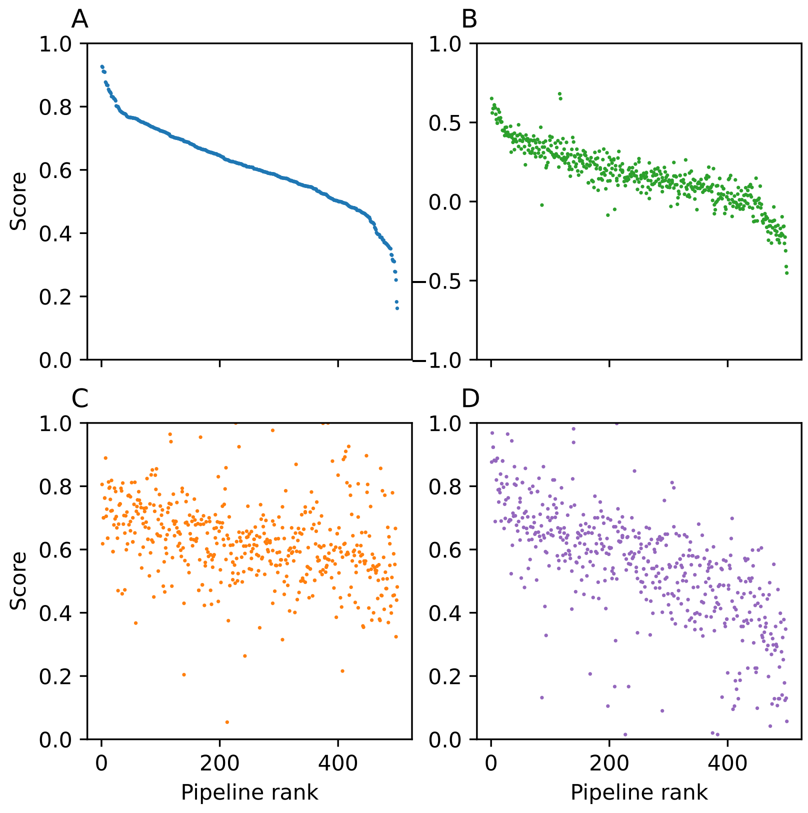

FIGURE 1.7: Classification statistics for the top 50 pipelines ranked according to AUC score; AUC (A), MCC (B), Specificity (C), Sensitivity (D).

The top 5 ranked pipelines discerned from the optimisation procedure are summarised in table 1.1 and table 1.2.

TABLE 1.1: Best performing pipelines with optimal processing steps and number of parameters n_θ.

Rank	Spectral smoothing	Baseline correction	Normalisation	Feature scaling	Feature extraction	Classifier	n_θ
1	N/A	N/A	Amide I	Robust	N/A	LR	1
2	SG	N/A	Min-Max	Standard	PCA	LR	3
3	N/A	N/A	Vector	Robust	N/A	LR	1
4	N/A	N/A	Amide I	Robust	N/A	LR	1
5	N/A	RB	Vector	Standard	PCA	LR	2

TABLE 1.2: Top ranking pipeline classification scores as decimals.

Rank	TN	FP	FN	TP	AUC
1	0.51	0.16	0.11	0.22	0.63 ± 0.02
2	0.47	0.20	0.12	0.21	0.62 ± 0.02
3	0.44	0.22	0.10	0.24	0.61 ± 0.02
4	0.45	0.22	0.09	0.25	0.61 ± 0.02
5	0.42	0.25	0.11	0.22	0.61 ± 0.02

Table 1.1 summarises the specific methods used in each of the top 5 ranked pipelines. **??** shows the distribution of performance measures aggregated over the 50 train-test splits for pipeline 1 and 2.

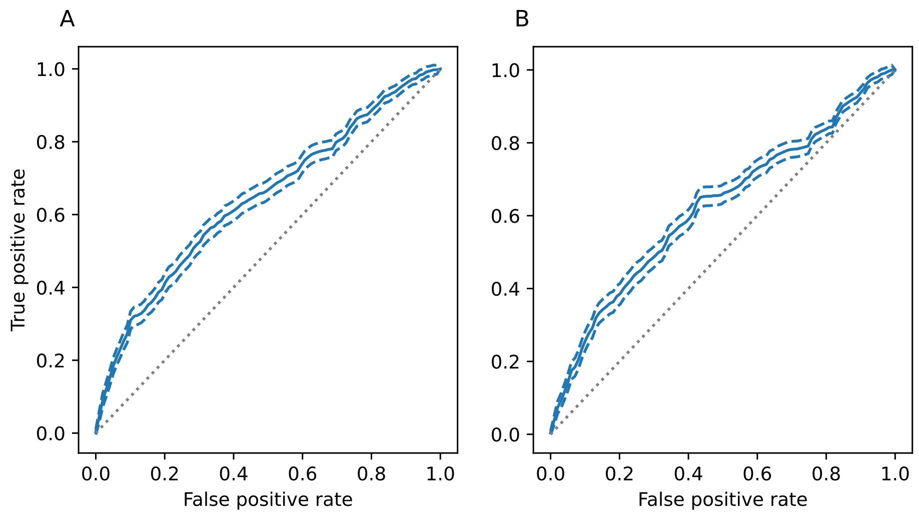

FIGURE 1.8: ROC curves shown with standard errors for best (A) and second-best (B) pipelines.

Figure 1.9 shows the loss functions sampled during the hyperparameter search for pipeline two. The loss surface is shows a strong dependence upon the fraction of components used in the feature extraction step. This contrasts with the relatively low dependence upon the regularisation parameter associated with the logistic regression classifier. This is likely due to the fact that both parameters play a regularising role in the inference procedure so as to avoid overfitting. If both steps were to have parameters indicating a high regularisation effect, this would be detrimental to the classification performance so feature extraction seems to be preferred.

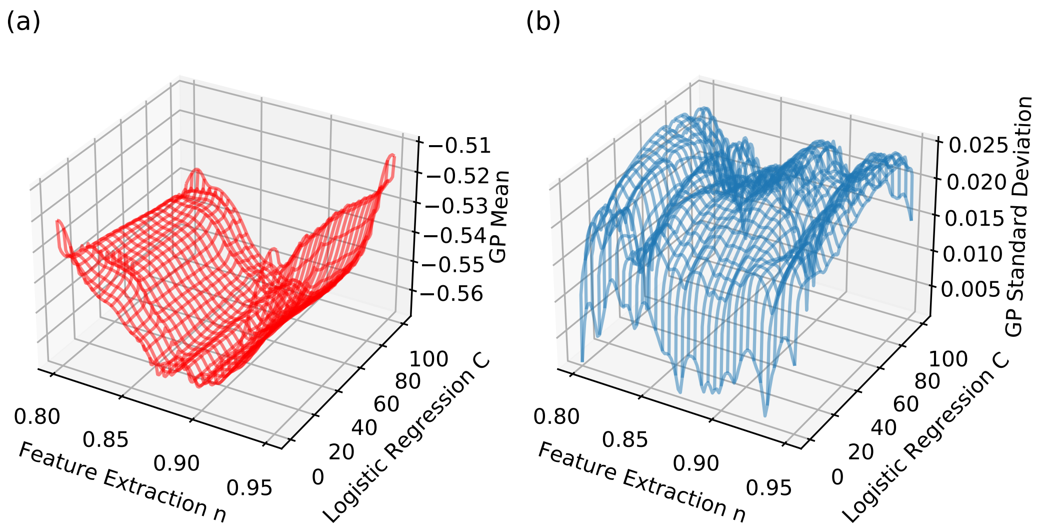

FIGURE 1.9: GP hyperparameter surfaces showing mean function in red and standard deviations in blue averaged across 50 sample iterations.

Optimised pipelines from each of the 50 train test splits each have a unique set of hyperparameters; these parameters are ”tuned” to specific training and validation data sets during the training phase. To determine a more generally applicable set of parameters the mode of the distribution of values was taken. Figure 1.10 shows histograms of each of the selected hyperparameters for the top two pipelines over 50 samples.

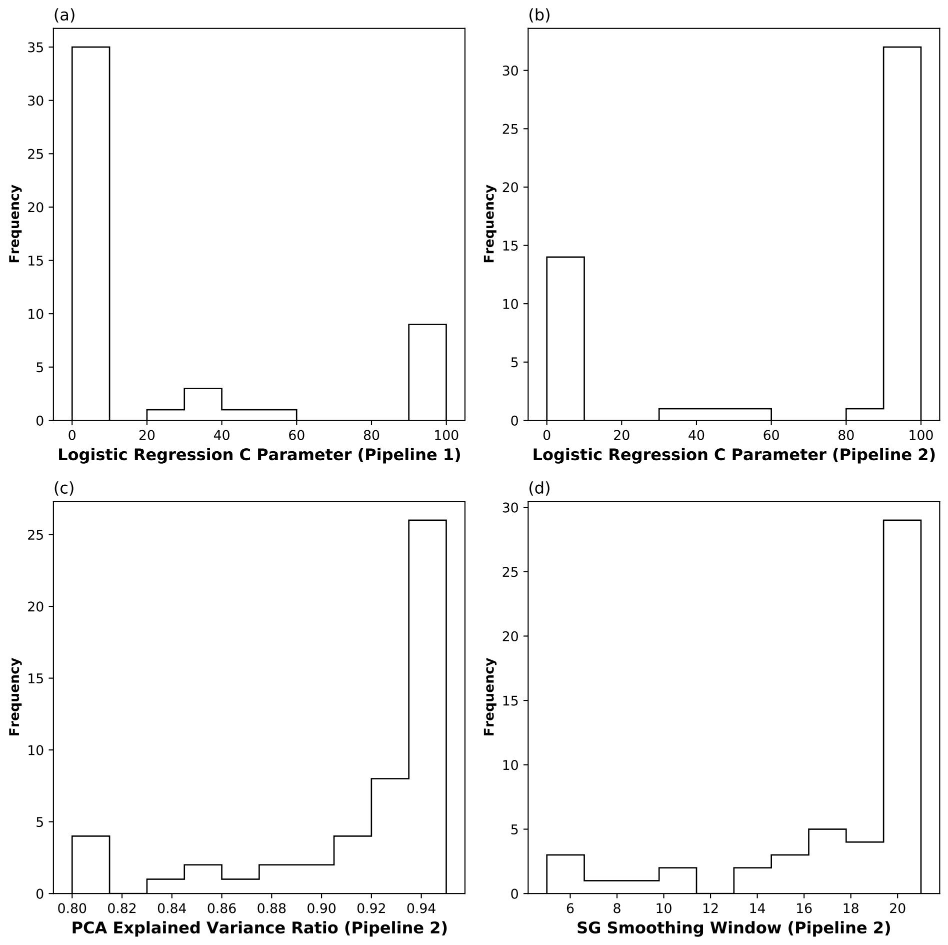

FIGURE 1.10: Histograms of optimum hyperparameters over the 50 train-test splits.

In order to acquire a more complete measure of the performance of the optimised pipelines, the model is sequentially tested 50 times by randomly drawing train-test splits without replacement. Modal values from fig. 1.10 are used as final model hyperparameter values. Aggregated statistics for pipelines 1 and 2 are shown in fig. 1.11.

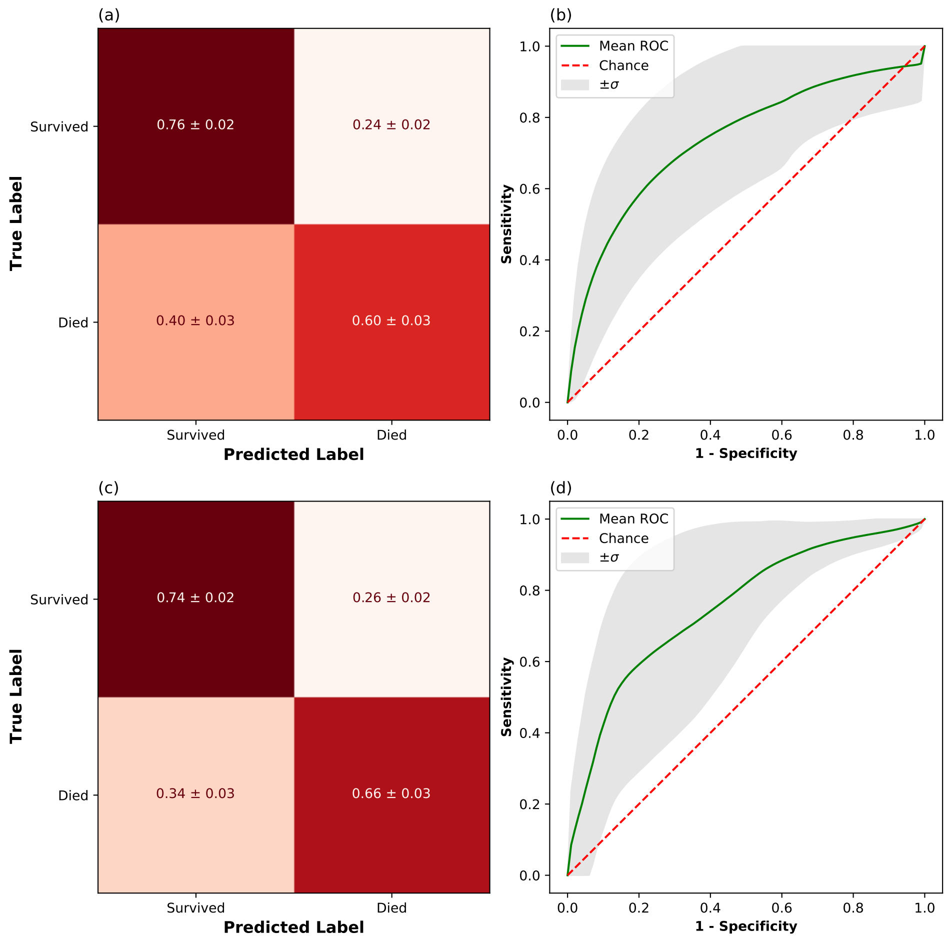

FIGURE 1.11: Mean confusion matrix and ROC curve shown with standard errors for best (a,b) and second best (c,d) pipelines trained and tested on full dataset.

Figure 1.11 shows a significant increase in both sensitivity and specificity when the optimised pipelines and hyperparameters are deployed on the full dataset. A lower specificity translates to a larger number of false positives indicating the model struggles to identify patients with a poor prognosis.

1.5 Discussion

classification scores vary widely but generally exhibit good classification scores well above random chance; shown in fig. 1.7. The more holistic measures of AUC and MCC follow a similar trend, whereas the relatively noisy sensitivity and specificity traces imply that there is often a trade off between the two metrics, where high sensitivity often leads to low specificity. Ranking metrics by AUC or MCC favours pipelines with balanced sensitivity and specificity scores. The trace in fig. 1.7(A) shows a number of small, relatively high scoring pipelines, before gradually decreasing towards an AUC of around 0.4, and an MCC of 0.0. The steep increase in AUC and MCC scores towards the higher end imply the expense of the optimisation procedure is justified.

Table 1.1 indicates that the optimal classifier for this dataset is logistic regression, with various choices of preprocessing options preceding this step. Normalisation and scaling are never bypassed, suggesting this is an imperative step. Two instances in the top five classifiers utilise PCA to reduce dimensionality, suggesting this step is not such an important for this dataset paired with logistic regression. Similarly, spectral smoothing by Savitzy-Golay filtering appears in the second pipeline, but is absent for the top ranking and remaining pipelines in the top five.

In order to investigate the effects of different methods on the performance of the pipeline, the frequency that a certain method either enhances or diminishes performances relative to a reference can be informative. Here, the reference score is the median score of all pipelines in the analysis.

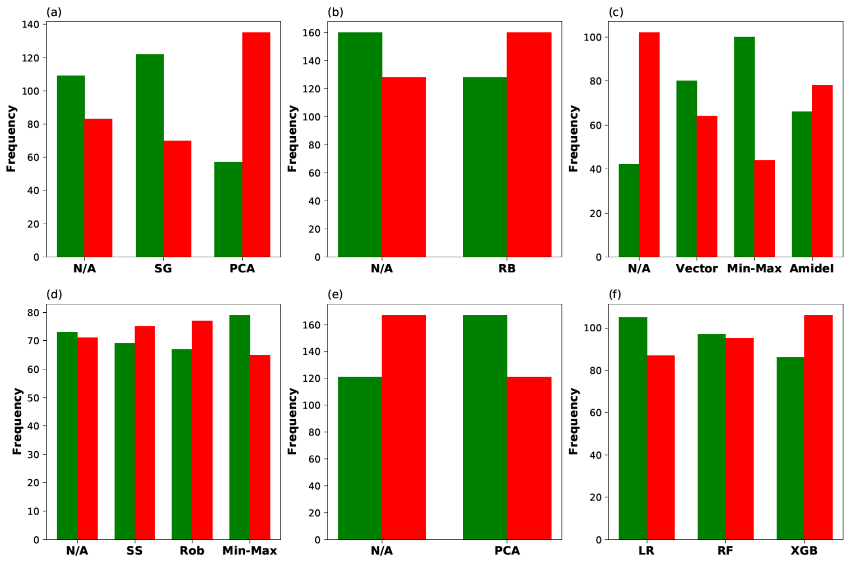

FIGURE 1.12: Frequency each method either enhances (green) or diminishes (red) relative to the median score (AUC = 0.48). Steps are (a) smoothing, (b) baseline, (c) normalisation, (d) scaling, (e) feature-extraction, (f) classifier.

Figure 1.12 shows some interesting insights of the effects of various methods. The choice of smoothing method evidently has a significant effect, the majority of pipelines which utilise PCA de-noising perform worse than the median, whilst Savitzy-Golay smoothing predominantly increases scores. It could be argued that baseline correction has an insignificant effect, perhaps slightly detrimental, this could be attributed to the data already being subject to a previous scatter correction prior to the analysis, negating the requirement to perform a baseline correction. Normalisation is evidently a step that can not be bypassed, an expected result as spectra originate from different samples each with differences in sample thickness. Mitigating the effects of sample thickness clearly has a positive effect on classification scores.

It appears that min-max normalisation occurs most frequently in the higher performing pipelines. Scaling of the data appears to have a significant effect on the performance of the pipeline, but the choice of scaling does not seem to be hugely important. It should be reiterated that the top five pipelines in table 1.1

employ a scaling method to the data in the pipeline, suggesting that re-scaling each wavenumber variable is beneficial for logistic regression. It would also appear that application of PCA to decompose the data prior to classification is slightly more beneficial than not. As previously stated, logistic regression emerges as a favourable classifier to the tree based random forest and gradient boosted classifiers, implying that a simpler, linear based model is preferred to complexity, perhaps as complex models are more prone to overfitting and have a much larger hyperparamater space to optimise. In fact, the dramatic drop off at approximately AUC = 0.40 is the result of pipelines with an XGBoost classifier, which has a large hyperparameter space that requires fine tuning. It may be the case that more iterations within the Bayesian hyperparameter search would produce more favourable results for the tree based models such as RF and XGBoost, but this would increase the time taken for the optimisation to execute.

The histograms in fig. 1.10 show distributions of hyperparameters for each of the two top performing pipelines. Interestingly, it reveals that the logistic regression regularisation value in pipeline one fig. 1.10(A) converges to a much lower value (~ 0.01) than for pipeline two, where it appears to converge towards 100. Pipeline two applies smoothing and feature extraction in addition to normalisation and scaling, which themselves have a regularisation effect on the subsequently fitted models. This may be the reason as to why the ultimate C parameter of logistic regression needn't be as low as 0.01 for pipeline two.

Taking the modal hyperparameter selections from fig. 1.10 and training and testing on all available data (using the same patients for each of the 50 train-test splits) enhances the scores significantly, as shown by the mean confusion matrices and ROC curves in fig. 1.11. For pipeline one, there is a 14% increase in mean specificity, and a 3% increase in mean sensitivity. Pipeline two exhibits an 11% increase in specificity and 9% increase in sensitivity. This would suggest that the strategy of sampling equally small subsets of data from each patient for the purposes of efficiency and stratification is sound, and translates well to a more realistic scenario where the all the available data from different patients should be used.

1.6 Conclusion

The work presented here demonstrates a versatile framework capable of determining a near optimal data preprocessing and classification pipeline. This optimization framework has been employed on a real inference problem and has successfully demonstrated that this process can be performed objectively and without specific prior knowledge of optimal parameters. The performance of the framework has been tested across a range of sample datasets and has shown that effective configurations can be determined through a rigorous analysis as proven by validation on held-out data. The choices of preprocessing methods resulting in pipelines with the highest ranks seem to follow conventional logic — normalisation is necessary, Savitzky-Golay smoothing is beneficial, PCA is beneficial depending on the choice of classifier. Valuable insights have been gained from the procedure showing that some preprocessing methods are particularly beneficial compared to others.

To gain further knowledge of effective preprocessing methods, it would be useful to perform the optimisation procedure on a wider variety of datasets. This could yield insights into which preprocessing steps are effective or given classes of inference problems.

This framework could be utilised by other researchers to perform a similar process for a given problem and set of preprocessing steps. It is by no means limited to FTIR spectroscopy and could be extended to other inference problems with minimal adjustment.

Bibliography

[1] Peter Lasch. Spectral pre-processing for biomedical vibrational spectroscopy and microspectroscopic imaging. *Chemometrics and Intelligent Laboratory Systems*, 117(May):100–114, 2012.

[2] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[3] David H. Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Computation*, 8(7):1341–1390, 10 1996.

[4] Roger M. Jarvis and Royston Goodacre. Genetic algorithm optimization for pre-processing and variable selection of spectroscopic data. *Bioinformatics*, 21(7):860–868, 2005.

[5] Elon Correa and Royston Goodacre. A genetic algorithm-Bayesian network approach for the analysis of metabolomics and spectroscopic data: Application to the rapid identification of Bacillus spores and classification of Bacillus species. *BMC Bioinformatics*, 12, 2011.

[6] Holly J Butler, Benjamin R Smith, Robby Fritzsch, Pretheepan Radhakrishnan, David Palmer, and Matthew J Baker. Optimised spectral pre-processing for discrimination of biofluids via ATR-FTIR spectroscopy. *The Analyst*, 2018.

[7] Abraham Savitzky and Marcel J.E. Golay. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 1964.

[8] Brian C. Smith. *Fundamentals of fourier transform infrared spectroscopy, second edition*. 2011.

[9] R O Duda, P E Hart, and D G Stork. Pattern classiïňĄcation. *New York: John Wiley, Section*, 2001.

[10] Isabelle Guyon and Andre Elisseeff. Feature Extraction, Foundations and Applications: An introduction to feature extraction. *Studies in Fuzziness and Soft Computing*, 2006.

[11] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize/scikit-optimize. 9 2020.

[12] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian process for machine learning*. The MIT Press, 2006.

[13] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The Condor experience, 2005.