Erasmus University Rotterdam

Erasmus School of Economics

Master's Thesis: FEM21031

Business Analytics and Quantitative Marketing

Feature Generation with Signal Processing Methods for Predicting Company Default

Author: Jan Bargeman, 538503

University supervisor: dr. Mikhail Zhelonkin

Second assessor: prof.dr. Dick van Dijk

Company supervisor: dr. Daniel Timbrell

August 14, 2021

Abstract

Data science has come to play a large role in the credit risk modelling of banks. This study aims to introduce a new application of signal processing feature generation from transaction data and use this to predict whether companies will default. The analysis is performed on private data from a major Dutch bank and on public data from the PKDD'99 Discovery Challenge. The performance of basic generated features is compared to signal processing methods like Fourier Transform and Wavelet Transform in various forms. Several mother wavelets are investigated for the Wavelet Transform, where the Daubechies 2 wavelet has the best results. On top of that, transformations of the data, being PCA, ICA and normalization, are also used to generate signal processing features from. Feature selection is then performed using recursive feature elimination and cross validation, where Shapley values are used as feature importance. Prediction is done with two machine learning classification algorithms, being LightGBM and Random Forest. Finally, for comparing the models, 5x2-cv with AUC followed by a t-test and McNemar's test are used. The results indicate that the signal processing methods significantly improve over the basic features on both data sets with both algorithms. Performance gains between 2% and 11% are found, depending on the data set and algorithm, which are all significant. The out of time performance of the private data is also tested, showing an expected slight performance drop, which is similar among features sets and algorithms. This study not only introduces another successful application of signal processing features but will also have a certain influence on how the financial industry performs their risk modelling. On top of this, it presents a practical implementation of this research in the form of an open-source package.

Contents

Chapter 1

Introduction

Large banks perform many types of credit risk modelling in order to assess, for example, a company's probability of defaulting on a loan the bank might provide. To successfully model this defaulting, many types of models are used, which are trained on large amounts of data. Recently, transaction data has been of interest, which consists of detailed information on each transaction from each client of a bank. From this data features can be generated that extract information in specific ways to create new variables with predictive power. However, generating a large number of features quickly results in impractical data sizes. On top of that, not all features have predictive power and for this reason feature selection is used to reduce the number of features, capturing only the most informative ones. These features are then used by machine learning algorithms to predict whether a company will likely default on a loan or not. This process assists in deciding whether the bank should provide this loan.

Every year approximately 350 billion in loans is provided by banks to Dutch companies[1]. The above explained approach is the benchmark way of working within one of the major Dutch banks. However, their feature generation stays within the more basic domain as they only generate simple summary statistics from transaction data over a recent period before the loan application. These summary statistics are: minimum, maximum, average, standard deviation, kurtosis, skewness and sum. With these models impacting such large sums of money, even a slight improvement in the model results in large gains.

The main focus of this thesis is a relatively unexplored technique in the field of financial data science: features generated with signal processing methods. It has had successful applications in healthcare regarding heart monitoring, in Lee et al. (1999), and recording of brain activity, in Prochazka et al. (2008). Furthermore, it was shown useful for electrical power systems, in Kang et al. (2010), automated vehicles, in Bilik et al. (2019), and speech recognition, in Stein (2002). The financial domain, however, has remained relatively untouched. Due to the high frequency of transaction data, signal processing has the potential to be very effective for information extraction. On top of that, looking at defaulting from a psychological point of view, indicates that being a steady payer decreases the chances of defaulting, (Liberati & Camillo, 2018). Signal processing also plays a role in this, as it assesses the steadiness of a signal and its persistence. The aim of this research is to apply these techniques to transaction data and compare it to the benchmark way of working within a major Dutch bank. The analysis is performed on private transaction data from this bank, as well as on a public data set with Czech transaction data from 1999.

[1] `https://www.statista.com/statistics/1133418/bank-loans-in-the-netherlands-by-sector/`

1.1 Research questions

To structure the research, the following research questions are established:

- What features can be generated from transaction data with signal processing methods that are informative to predict whether companies default?
- How to reduce the number of features in order to keep the data size manageable?
- How do these features perform compared to the benchmark way of working inside a major Dutch bank?
- How do these features compare regarding scale-ability for practical implementation?

From these research questions the approach to the problem becomes clear. First, various types of signal processing features, like Fourier Transform and Wavelet Transform, will be generated from the transaction data. On top of this, these features will also be generated from PCA, ICA and normalized versions of the data sets. Then, feature selection is performed, using Shapley values as feature importance, to capture the most informative features. Finally, the performance of the selected set of features is compared to the performance of the benchmark features. This is done with LightGBM and Random Forest. 5x2-cv followed by a t-test and McNemar's test are used as statistical tests to indicate whether the performances are significantly different. Furthermore, to investigate whether this feature generation method is practical for the banks, the feature generation and model training times are collected.

1.1.1 Results preview

Following this above described approach, the analysis concludes that for both data sets the inclusion of signal processing features significantly improves the performance of the models. For the private data set the performance improves from 0.69 AUC to 0.80 AUC with LightGBM and from 0.71 AUC to 0.73 AUC with Random Forest. For the public data set the performance improves from 0.91 AUC to 0.98 AUC with LightGBM and from 0.89 AUC to 0.94 AUC with Random Forest. The features from PCA, ICA and normalized versions of the data sets did not significantly improve the performance for either data set or algorithm, but did lower the standard deviation of the cross-validated performance. The feature generation times approximately double and the model training times increase with approximately 50%, but this is acceptable for such a sizeable improvement in performance. This research concludes that signal processing features are certainly valuable for credit risk modelling.

1.2 Structure of the paper

The paper is organized as follows. Section 2 introduces the literature available on these topics, capturing the current state of research. Section 3 describes the data, its properties and the necessary preprocessing. Section 4 elaborates on the methods used for the feature generation, feature selection, machine learning and model comparison. Section 5 presents the results, consisting of an empirical study assessing the performance of these features and comparing the models. Finally, Section 6 gives a conclusion.

Chapter 2

Literature

In this section, the existing literature is reviewed. Relevant outcomes and trends are presented while indicating which areas are still left untouched. Furthermore, the choices for this research are explained, where for relatively arbitrary choices the standard method of the bank is preferred. The topics that are discussed are predicting default, feature generation, feature selection and model comparison. The section concludes with a summary.

2.1 Predicting default

Attempting to predict whether a company will default on a loan has a long history of research. García et al. (2015) summarize this and determines some notable characteristics. The most relevant are the imbalance in the data sets, with sometimes only 1% of defaults, and the asymmetric costs of false negative and false positive errors. Besides these, they also mention that data sets can be noisy, with atypical observations clouding the underlying relationships. García et al. (2015) also indicate that 68.8% of research in this field performs their analysis on only one database, and 13.5% on two databases, like this research.

2.1.1 Class imbalance

This previously mentioned large imbalance in the data, is called the Low Default Portfolio problem, (Khemakhem & Boujelbene, 2018). Some data sets only have 1% defaulting loans, which makes it arbitrary to fit a model that has 99% accuracy, only predicting non-default.

An intuitive attempt to solve this problem would be to over-sample the minority class with replacement. However, it has been shown that it does not necessarily improve minority class recognition, (Japkowicz, 2000; Ling & Li, 1998).

Similarly, under-sampling the majority class could also be a possible approach. Here, the original population of the minority class is kept and a set of observations is randomly selected from the majority class. In one experiment it is noted that the best performance is obtained when the classes are of equal size (Ling & Li, 1998).

Finally, a more involved method is called the Synthetic Minority Oversampling Technique (SMOTE) introduced by Chawla et al. (2002). The method generates extra synthetic training data by performing certain operations on the real data. In this case they take a random selection of k nearest neighbors from an observation in the minority class and calculate the distance between the feature space of this observation and the feature space of the random selection of nearest neighbors. Then, this distance is multiplied by a random number between 0 and 1, added to the selected observation and a new minority class member is created.

For practical reasons explained in Section 3.3, the preferred choice for handling the imbalance problem in this research is undersampling the majority class.

2.1.2 Classification

When it comes to predicting whether one will default or not based on explanatory variables, there are generally two main approaches. The first is statistical methods like multivariate discriminant analysis, (Altman, 1968), or (logistic) regression analysis, (Ohlson, 1980; Hillegeist et al., 2004), which are both often used as standard.

However, Kruppa et al. (2013), Trustorff et al. (2011) and Barboza et al. (2017) all show that nowadays these traditional statistical methods are not as efficient or accurate for credit risk classification as the second approach: machine learning techniques. These techniques are algorithms like Support vector machines, (Cortes & Vapnik, 1995), Random Forest, (Breiman, 2001), decision-tree boosting, (Schapire, 2003), or artificial neural networks, (Gurney, 2018). Some researchers go even further, suggesting that ensemble methods of multiple machine learning algorithms outperform every stand-alone classifier (Nanni & Lumini, 2009).

This mentioned reduction in efficiency is partially due to the number of explanatory variables being very large and machine learning techniques being more efficient in handling this. For this research, many thousands of features will be generated, so machine learning methods are preferred. In an extensive study on the default predicting performance of these algorithms, Barboza et al. (2017) found that the Random Forest performed best, agreeing with Kruppa et al. (2013). For this reason, Random Forest is chosen as the classifier for this research, together with LightGBM because it is the standard within the bank. LightGBM, introduced by Ke et al. (2017), is a state-of-the-art classification boosting algorithms which excels at the previously mentioned imbalance problems. Both of these algorithms are further explained in Section 4.5

2.2 Feature generation

Machine Learning methods almost always perform better when extracted information from the original data is also used for training the model, (Nargesian et al., 2017). For example, explicitly feeding an algorithm what the lowest yearly balance is for each account, will likely prove informative for predicting whether one will default. This idea can be broadened to, for example, the maximum, the standard deviation or skewness of their transactions. In this way the algorithm receives a more qualitative description of the raw input data, which generally boosts the model performance.

Most feature generation in the field of classification for risk modelling generally stays within the more basic domain. An example is Khandani et al. (2010), who aggregate the data over different time window lengths, using mainly sum and average, or Härdle et al. (2009) who calculate some basic indicators for profitability, liquidity, etc. This indicates that there is room for complementary research in this domain.

2.2.1 Signal processing

Signal processing is used successfully for feature generation in several other fields like medicine and electrical engineering, as discussed in Chapter 1. However, in the financial domain it is scarcely used. One example is stock forecasting by feature generation from image processing of charts, by Du et al. (2020). They apply a Wavelet Transform with the Daubechies wavelet (db4, Daubechies & Sweldens (1998)) as their mother wavelet to extract frequency information from stock charts. Mother wavelets are further discussed in Section 4.2. However, it is clear that

the classification for risk modelling area has not used signal processing for feature generation yet.

Another interesting angle is preprocessing the data before applying signal processing methods. This has been done by Zhang et al. (2015), who use Principle Component Analysis (PCA) on the data before using a Fourier Transform to perform facial recognition. Furthermore, transforming the data with PCA and Independent Component Analysis (ICA) is also done before a Wavelet Transform in the field of medicine, (Martis et al., 2013; Giri et al., 2013). This approach will also be performed in this research and a more detailed explanation and methods for implementation for both PCA and ICA are given in Section 4.3.

2.3 Feature selection

Feature selection is the process of reducing the input data for an algorithm by removing uninformative features, (Liu & Yu, 2005). Methods like regularized logit use Lasso and Ridge regularization, also known as shrinkage methods, to reduce or shrink the coefficients in a logistic regression, (Tibshirani, 1996). Other researchers suggest looking at the feature values, with for example large Wavelet Transform coefficients being a proxy for a persistent signal, (Nayak & Panigrahi, 2011). Another way of doing this is recursive feature elimination, as suggested by Guyon et al. (2002). This method calculates the informativeness, or importance, of each feature with the use of a machine learning algorithm. After this, a certain number of uninformative features is removed and the process is repeated.

One way of computing these feature importances is by using an algorithm like Random Forest that has built in feature importance, (Altmann et al., 2010). These methods use for example the mean decrease in accuracy or Gini coefficient, (Dorfman, 1979), of each feature or simply count how often it is used to split the decision trees. Another type of feature importance comes from game theory and is called the Shapley value, introduced by Shapley (1951). A player's contribution is calculated with

$$\phi_i(v) = \frac{1}{p} \sum_{S \subset P \setminus \{i\}} \binom{p-1}{|S|}^{-1} (v(S \cup \{i\}) - v(S)), \tag{2.1}$$

which can be interpreted as

$$\phi_i(v) = \frac{1}{\text{number of players}} \sum_{\text{coalitions excluding } i} \frac{\text{marginal contribution of } i \text{ to coalition}}{\text{number of coalitions excluding } i \text{ of this size}}, \tag{2.2}$$

where, in Equation 2.1, p is the total number of players, P is the set of all players, S is each possible coalition formed without player i and $v()$ is a characteristic function describing the worth of the cooperation by coalition S. By iterating over every coalition S and summing the marginal contribution of player i to that coalition, that player's total contribution, or importance, is found. This entire method can be cross-validated and is instructively implemented by Dell'Agnola et al. (2020). On top of this, this method is also the standard method for the bank. The choice of feature selection method is relatively arbitrary, and thus the chosen method is recursive feature elimination by using Shapley values. A detailed description of the implementation is given in Section 4.4.

2.4 Model comparison

2.4.1 Performance metric

To compare different models, first a performance metric is chosen. García et al. (2015) shows that 85% of research in the field of predicting default uses, under more, accuracy as their performance metric, while 40% uses type-I/type-II error and 15% uses AUC. Though Section 2.1 already mentions the asymmetric penalties, the actual cost of misclassifying a defaulter or non-defaulter is only used by 5%. This is likely because the exact losses or profits are generally not available. This is also the case for this research as neither the public nor private data set contain this information. Because of these asymmetric penalties, García et al. (2015) suggests that accuracy does not seem to be an appropriate metric, as it assumes equal misclassification costs. For this reason, they propose AUC to be the more appropriate performance measure. On top of that, AUC is also the preferred metric by the bank, hence it is for choice. The concept behind AUC is further explained in Section 4.6.1.

2.4.2 Performance comparison

Then, to compare the performance of a number of features sets, an algorithm is trained on these sets to produce an equal number of models. Then, the performances of these models are compared by using a test set. Specifically for comparing sets of features, model comparison is generally done using ten-fold cross-validation, (De Chazel & Reilly, 2000; Kahya et al., 2006; Ayata et al., 2016). However, statistical hypothesis testing is not used by these researchers. Dietterich (1998) gives several suggestions regarding statistical tests for comparing classification algorithms, from which McNemar's test and 5 times 2-fold cross validation (5x2-cv) are used for this research. Both of these are more elaborately explained in Section 4.6.2.

McNemar's test is suggested as the statistical hypothesis testing method when computation time is an issue, which it might be as discussed in Section 3.3. 5x2-cv followed by an independent t-test has a similar low false positive rate and is implemented when data size is moderate by Cieslak & Chawla (2008) and Bouckaert (2003). This test provides higher power at the cost of 10 times more computation time. Finally, Nadeau & Bengio (2003) does have criticism on these tests suggested by Dietterich (1998), as they say they lack taking into account the variability problems due to overlap of training sets for the cross-validation process. However, they conclude that correcting for this does not considerably influence the results from these tests.

2.4.3 Out of time validation

Finally, specifically for time series data, Stein (2002) proposes a new approach to model validation. By selecting an early time window, training the algorithm on this and then validating on the next time window, the predictive intention is used throughout the validation process. This method is repeated on all following time windows, till the last one. This final time window is then used as an Out Of Time (OOT) holdout sample. However, due to the nature of the used data, discussed more extensively in Section 4.6.3, this approach is not feasible. As explained in that section, a regular OOT holdout sample will be kept from the private data.

2.5 Summary

This research intends to investigate whether signal processing features improve the predictive performance of machine learning algorithms. In this way a valuable contribution is made to the domain of feature generation in finance. For feature selection, recursive feature elimination using

Shapley values will be used, while using cross-validation throughout the process. Furthermore, it compares the renowned Random Forest with the state-of-the-art LightGBM algorithm, assessing whether either algorithm profits more from these features than the other. Finally, both 5x2-cv and McNemar's test will be used to statistically assess relative performances of the features sets with the AUC metric.

Chapter 3

Data

In this section, the data sources and its properties are discussed. Furthermore, the necessary preprocessing before applying signal processing methods is explained. Finally, the practical implementation regarding the handling of the imbalance is discussed.

3.1 Sources and properties

The main type of data for this research is transaction data, which is a log of client accounts with all of their transactions from a certain time period. Besides this, it generally also contains their balances on each date of the time period. An example of the transaction and balance data of an account from the public data set is given in Figures 3.1 and 3.2, respectively. For this research, plain transaction data is not enough. The data should contain a label that is to be predicted based on this transaction data. In the case of this research the preferred label will be the status of a loan or a credit rating.

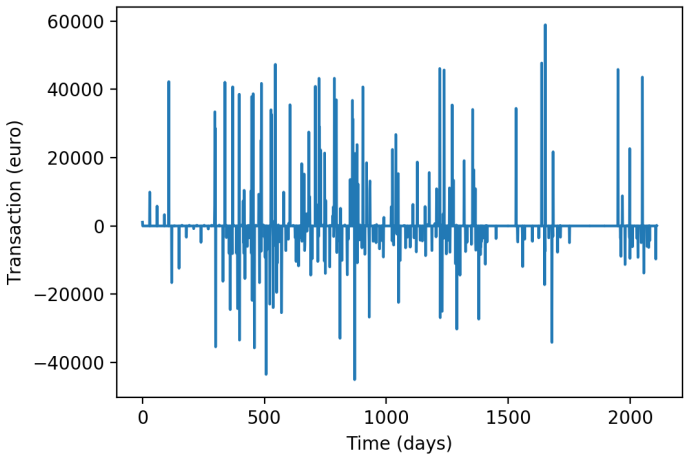

Figure 3.1: Example of transaction data from one account from the public data set.

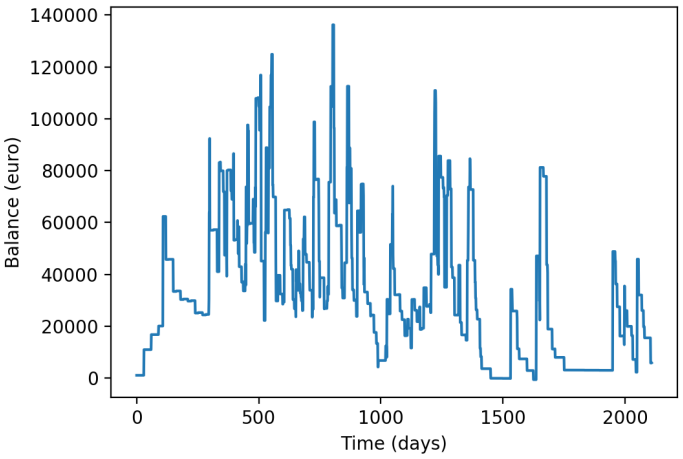

Figure 3.2: Example of balance data from one account from the public data set.

3.1.1 Private data

The first data set is provided by a major Dutch bank and contains real transaction data from companies from their Belgian division, spanning from July 2016 till April 2019. In total it consists of 73,672,986 transactions from 47,416 accounts. These accounts received a total of 196,582 loans, out of which 3,636 (1.8%) defaulted. The label to be predicted is the status of

the account's loans, with 1 being default and 0 being non-default. This data is private and can not be found publicly, as this thesis is part of an internship.

3.1.2 Public data

The second data set comes from the data science website Kaggle. It hosts various competitions regarding data processing and modelling and contains publicly available data sets. The used data[1] was publicized in 1999 during a Principles of Data Mining and Knowledge Discovery conference in the Czech Republic. It contains real anonymized Czech bank transactions and loan info from 1993 till 1998. The label to be predicted is the account's credit rating at 31-12-1998, ranging from A to D, with the lowest rating (D) being default. To make the label binary, the credit ratings A through C were replaced by 0 (loan status = non-default), and D by 1 (loan status = default). It consists of 191,556 transactions from 682 accounts, out of which 45 (6.6%) defaulted.

3.2 Preprocessing

As is more extensively explained in Section 4.2, signal processing methods require a constant sampling frequency. The transaction timestamps from the private data set are detailed to the minute, but the public data only has days. Therefore, a daily sampling frequency is chosen. This means that each daily timestamp should have a value to be processed. As the data sets contain only values at timestamps when a transaction is made, all other timestamps have to be filled in. Transactions are filled with 0, because at empty timestamps there are no transactions. Balances are forward filled, meaning that the balance value is repeated until a new value occurs, because no transaction means no change in the balance. Though there is no way of recognizing missing transactions, this processing of the data implies that there are no missing values. In the data it also happens that multiple transactions occur on the same day. In this case they are combined, as signal processing requires there to be only one value at each sampling timestamp. To combine these observations, transactions are summed and the final value of the balance is taken.

3.3 Handling the imbalance

As discussed in Section 2.1 and seen in the data properties discussed above, there is a large imbalance. There are several approaches to this, however, one stands out from the rest regarding practical implementation. Because this research intends to generate thousands of features, train a vast number of machine learning algorithms and perform extensive cross-validation, the computational expensiveness will be high. On top of this, the private data is contained on a high security cloud environment, which is relatively slow and occasionally unreliable. By using majority undersampling, which means taking all defaults and an equal number of randomly sampled non-defaults, as suggested by Ling & Li (1998), this speeds up the feature generation considerably and removes the imbalance problem. To account for this choice, it is reiterated that the goal of this research is to identify the behaviour of the defaulters, as opposed to the behaviour of the non-defaulters. By taking all defaults, their patterns are fully represented in the analysis.

[1] https://www.kaggle.com/pranati25/predict-loan-defaulters

Chapter 4

Methodology

In this section, the methods used in this research are explained. First, the approach regarding feature generation, along with signal processing and the transformations, is presented. Then, feature selection and the different classification algorithms are discussed. Finally, the methods for comparing the different models and the benchmarking are given. Each section has a dedicated subsection discussing the practical implementation.

4.1 Feature generation

As explained in Section 2.2, feature generation is the process of taking the raw input data and extracting specific information from it, called features. In this research, there is a divide between basic features and signal processing features. The basic features, called 'Basic' in Results, consist of simple summary statistics, representing the benchmark way of working within the bank. In this case these are: minimum, maximum, average, standard deviation, kurtosis, skewness and sum. The signal processing features are generated with more involved methods, which are discussed in Section 4.2.

4.1.1 Time windows: implementation

Feature generation can be performed on all data or on selected time windows of preference. Taking into account that the data concerns companies, quarterly and yearly time windows are used. For quarterly, this means that a selected period, here one year, is split into quarters and for each quarter the features are generated. The same features are also generated over one year. The minimum duration of one year is chosen as it resembles real banks, who refuse to give loans to clients from whom they have too little data. Also, data beyond this year is discarded in order not to be biased regarding the customer relation.

The trailing twelve months before the loan application are used, disregarding which specific months these are. This results in different quarter lengths, depending on when the loan application is issued. Due to the need for machine learning algorithms to be able to compare features, the first 88 days of each quarter are used. This assures that there are no missing values in the features.

4.2 Signal processing

With signal processing, the goal is to extract a signal from the data. A signal is described as a recurring wave-like pattern, with properties like frequency, amplitude and phase, (Willsky & Young, 1997). For these methods to work, the data should be sampled at a constant interval, which is the sampling frequency. A problem for every signal processing method is that the

sampling frequency overpowers every other signal in the data. For this reason, the sampling frequency is filtered out, in accordance with Nyquist's criterion, as introduced by Nyquist (1932).

4.2.1 Fourier Transform

The discrete Fourier Transform is a technique which decomposes the data into a combination of sines with different frequencies, amplitudes and phases. The size value at a certain frequency is the amplitude of the signal, and the phase indicates where it starts. It is given by

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{\frac{i2\pi}{N} kn}, \tag{4.1}$$

which can be rewritten as

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N} kn\right) - i \cdot \sin\left(\frac{2\pi}{N} kn\right) \right], \tag{4.2}$$

where Equation 4.2 follows from Equation 4.1 by Euler's formula: $e^{ix} = \cos(x) + i \cdot \sin(x)$. Here N is the length of the data and k goes from 0 to $N - 1$. As is visually explained in Figure 4.1, it transforms the data from the time domain into the frequency domain. In the figure it is seen that a signal, in red, is decomposed into a series of sines, in purple. These sines each have a certain frequency, seen as the location of the spikes on the x-axis in the blue graph. The height of the spike represents the amplitude of the signal. Figures 4.2 and 4.3 show an example for the public data set, where a year of transaction data returns a large value, or amplitude, for the frequency 12 and resonance at multiples of this frequency. This indicates that the data exhibits steady large changes 12 times per year. An interpretation could be, for example, monthly salary payments.

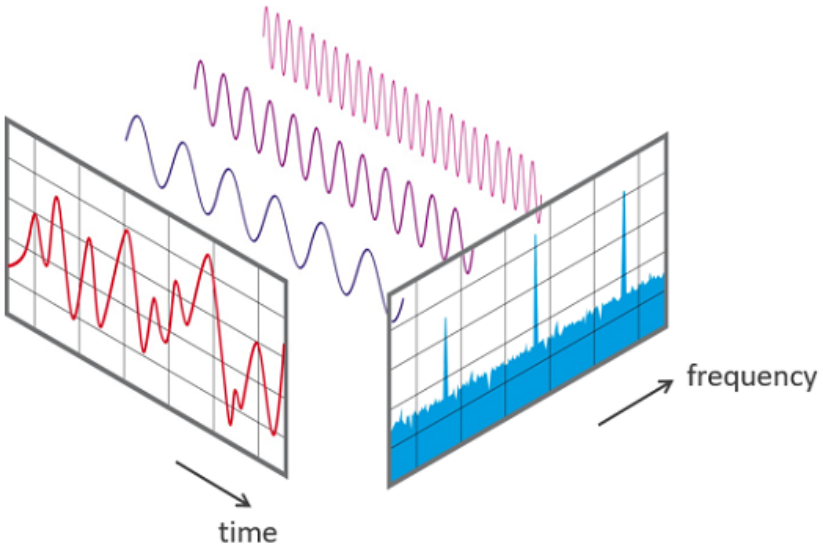

Figure 4.1: A visual representation of the Fourier Transform. Reprinted from NTi-Audio.com[1], by NTi-Audio, 2017

[1] https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

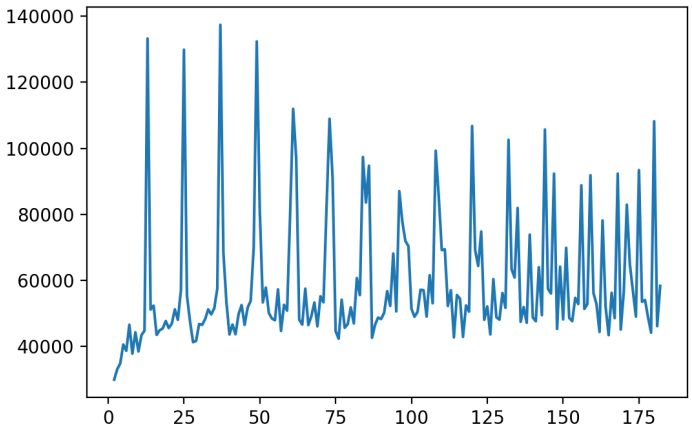

Figure 4.2: Averaged Fourier Transform representation of one year of transactions from the public data.

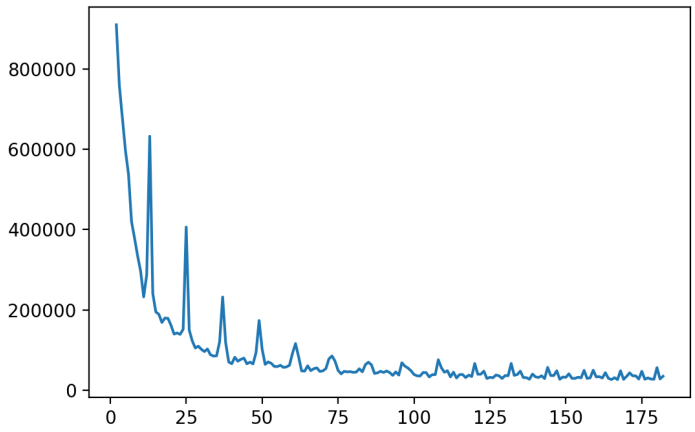

Figure 4.3: Averaged Fourier Transform representation of one year of balances from the public data.

Fourier Transform: implementation

For the Fourier Transform the SciPy (Scientific Python) package[2] with default parameters is used and it is implemented in two ways. For the first implementation, the complete Fourier representation is used, called 'Fourier complete' in Results. This means that for a quarter, with data length 88 as discussed in Section 4.1.1, it filters out the sampling frequency and for the resulting 44 frequencies the amplitude and phase are used as features, adding in total 88 features.

For the second implementation, a selection of largest valued frequencies is used, called 'Fourier n-largest' in Results, where n is the number of largest valued frequencies to use. The larger the value at a certain frequency, the more present this signal is in the data. It is intuitive to select a certain number of largest valued frequencies and let the machine learning algorithms formally compare them. With for example n = 10, the frequencies and amplitudes of the 10 largest valued frequencies are used as features, adding in total 20 features.

4.2.2 Wavelet Transform

The discrete Wavelet Transform is another type of transform, which not only gives information on the frequencies, but also on where the signal occurs in the data. In this way it differs from the Fourier Transform, which only gives frequency information, but is more detailed. It is given by

$$\gamma_{jk} = \sum_{t=0}^{T} x(t) \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - k2^j}{2^j}\right), \tag{4.3}$$

where γ is the detail coefficient, ψ is the mother wavelet, j is the scale parameter and k is the shift parameter, with k and j both integer. Now, if for j a scale is chosen, so that γ_{jk} is only a function of the time shift k, then it can be seen as a convolution of $x(t)$ with the inverted mother wavelet. This means that a Wavelet Transform slides a short piece of wave, the mother wavelet ψ, past the data $x(t)$ and multiplies the nearby values with this wavelet. An example of such a mother wavelet is given in Figure 4.4. The result of sliding and multiplying the wavelet past the data is a representation of the data with respect to this wavelet, called the detail coefficients γ_{jk}. If the data has some wave-like characteristics resembling the mother wavelet, these details

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.fft.html`

coefficients will return high values. There are various shapes of mother wavelets, and these result in different representations of the data. The detailedness of the mother wavelet can also vary. By having a very detailed mother wavelet, the data should resemble the mother wavelet closely in order to return high values. The opposite is also true, as having a less detailed mother wavelet requires the data to only loosely resemble the shape of the wavelet. The way in which the Wavelet Transform retrieves frequency information, is that it decomposes the data into high and low frequency components, indicated by the scale parameter j. It then performs the same convolution with the mother wavelet again on the low frequency data. The low frequency components of the data are called the approximation coefficients. The data can be decomposed several times and this is called the depth. The maximum depth is dependent on the length of the data and the detailedness of the mother wavelet. This sliding can be performed at each depth, and will give a representation of the data at those frequency bands.

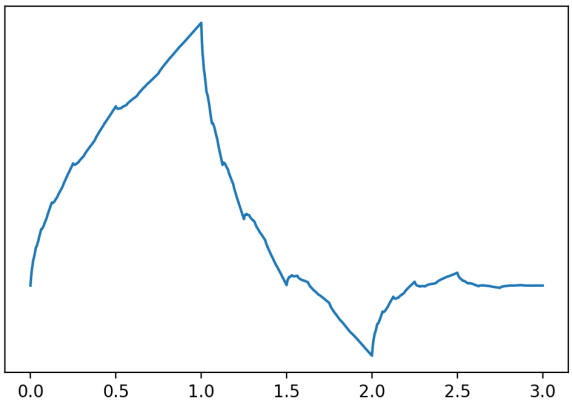

Figure 4.4: Example of a mother wavelet: Daubechies 2.

Wavelet Transform: implementation

For the Wavelet Transform the PyWavelets package[3], with default parameters except for the mother wavelet, is used and it is implemented in two ways. For the first implementation, the complete Wavelet representation is used, called 'Wavelet complete' in Results. The features are all approximation and detail coefficients at each depth.

For the second implementation, the basic features from Section 4.1 are generated from both the approximation and details coefficients at each depth. This means that at each depth 14 features are generated.

4.3 Transformations

In an attempt to boost the performance of the model and possibly reduce the number of final features simultaneously, several data transformations are considered. The first transformation is PCA, Principal Component Analysis. This is a popular statistical technique of condensing the information from many variables into fewer. The k largest eigenvectors of the covariance matrix, which still explain a considerable portion of the variance in the data, are selected, (Jolliffe, 2005). Such an eigenvector is a combination of the original variables that, together as one

[3] `https://pywavelets.readthedocs.io/en/latest/ref/dwt-discrete-wavelet-transform.html`

variable in this proportion, explain the variance in the data to a relatively high degree.

The second transformation is ICA, Independent Component Analysis. Though lesser known than PCA, it has a similar goal according to Draper et al. (2003). In contrast to picking the largest eigenvectors, it pursues to select maximally independent (orthogonal) vectors. These vectors attempt to distinguish the different signals present in the data.

Finally, normalization is used to transform the data. Normalizing the data before applying signal processing is often done, for example by Shaker (2006), who normalize their brain recording data before using a Fourier and Wavelet Transform. For normalization, the considered data is simply normalized using $X_{normalized} = (X - X_{minimum})/(X_{maximum} - X_{minimum})$. Doing this scales the analyzed data to a range between 0 and 1. This results in the shapes of the data being compared more fairly. Specifically, the signals in the data will be compared relatively more on shape than on amplitude.

4.3.1 Transformations: implementation

When generating the features with these transformations, the practical implementation is as follows. The transformation is applied to the considered data of each account in the specific time window, e.g., one quarter, one year. For PCA and ICA the balance and transaction data are transformed together into two PCA or ICA vectors, respectively. The used packages are PCA[4] and FastICA[5], both from scikit-learn. Feature generation is then performed with both of these vectors. For normalization, the considered data is normalized using the above given formula, after which the same feature generation is performed.

4.4 Feature selection

As explained in Section 2.3, feature selection is the process of removing uninformative features. In this research recursive feature elimination with cross validation is used. This is done using 5-fold cross validation and the performance metric is AUC. First, the model is trained on all features and the importance of each feature is calculated using Shapley values, which are more thoroughly explained in Section 2.3. Then, a certain number of least informative features is removed. Repeating the process with this new set of features, the performance of the algorithm is measured with each shrinking set of features. The performance does not drop immediately, as at first uninformative features, acting as noise, are removed. Finally, when the process starts to remove the informative features, performance of the algorithm begins to decrease, as seen in Figure 4.5.

4.4.1 Feature selection: implementation

Recursive feature elimination

The above process is implemented in the package ShapRFECV[7]. One needs to specify the amount of features to be removed at each iteration. In the case of this research, a step of 0.3 (remove 30% of current least informative features) is used to go from more than one thousand features to around four hundred. Then, a step of 0.2 is used to go to around 50 features. Finally, a step of

[4] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html

[6] https://medium.com/ing-blog/open-sourcing-shaprfecv-improved-feature-selection-powered-by-shap-994fe7861560

[7] https://ing-bank.github.io/probatus/api/feature_elimination.html#probatus.feature_elimination.feature_elimination.ShapRFECV

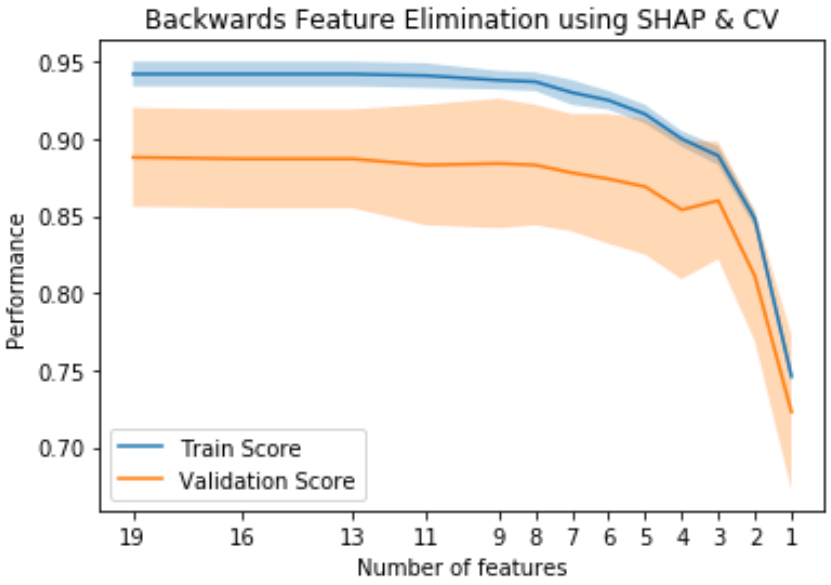

Figure 4.5: Typical example of recursive feature elimination. Reprinted from Medium[6], by Mateusz Garbarcz, 2020

0.1 is used to find the optimal number of features. This three stage process limits computational expensiveness, while still preserving a detailed result.

Features subsets

Feature selection is performed on three sets of features. The first set is called 'Basic set' in Results and contains only the regular basic features, without any signal processing or transformations of the data. The second set is called 'Regular set' in Results and contains the regular basic and regular signal processing features. Finally, the third set is called 'All set' in Results and contains all features, both basic and signal processing from both regular and transformed data.

4.5 Machine learning

Machine learning algorithms attempt to predict a label based on input data, where in this case the label is whether a client defaults or not. This is also known as a binary classification problem, with 1 for default and 0 for non-default. These algorithms solve this problem by processing input data to detect patterns and combinations which relate to the label. By explicitly feeding certain properties of the data, called features, to the algorithm, it generally performs better.

Some explanation regarding model training is relevant for the coming sections. An algorithm is trained on a selection of the data, called the training set. After this, the performance is gauged on the rest of the data, which it has not seen before, called the validation set. For this research a train/validation split of 70/30 was used throughout.

4.5.1 Classification algorithms

For this research, two types of machine learning algorithms are used. The first one is Random Forest and the second is LightGBM. Both will be discussed more in depth in the following sections.

Random Forest

Widely used in banking for its relatively explainable predictions, Random Forests are a large collection of decision trees, introduced by Breiman (2001). Each decision tree uses a random subset of variables and observations for training. It is possible to limit the number of decisions per decision tree and this is called the depth. After training the complete Random Forest, majority voting is used for classification of new samples. For this analysis the package RandomForestClassifier[8] by scikit-learn is used. All default parameters are used, except those given in Section 4.5.2.

LightGBM

LightGBM, or Light Gradient Boosting Machine, is a state-of-the-art gradient boosting tree algorithm as mentioned in Section 2.1. Boosting is the process of training several weak classifiers and combining them into a strong classifier. This is done iteratively, with each new classifier improving on the mistakes of the previous one. In general, boosting algorithms can suffer from overfitting, meaning that it is too dependent on the input data, as explained by Schapire (2003). The result is that it performs very well on the training set, but not well on the validation set. For this reason, boosting algorithms generally have a built-in validation system. In this validation system, a validation set size of 30% is used. For this analysis the package LGBMClassifier[9] is used. All default parameters are used, except those given in Section 4.5.2.

4.5.2 Grid search

Key part of machine learning algorithms is hyperparameter optimization. For example, for a Random Forest choices have to be made regarding the number of trees and their maximum depth to reduce overfitting. For the purpose of this thesis, a simple grid search will suffice. In grid search a reasonable hyperparameter space is defined, with for each parameter a discrete interval. Then, for each combination of hyperparameters, a machine learning algorithm is trained. The performance of each of these different algorithm implementations is measured with K-fold cross-validation, where K=5. The hyperparameters resulting in the implementation with the best average performance are used for the analysis.

An overview and explanation of the hyperparameter grid for both algorithms is given in Appendix A.1.

4.6 Model comparison

The goal of the models is to maximize the correctly identified number of defaults, while simultaneously minimizing the number of wrongly-accused non-defaulters. Model comparison is used to gauge the relative performance of the models. In this way it can be checked whether one model significantly outperforms another.

4.6.1 Performance metric

As discussed in Section 2.4.1, the measure of performance for this research is the Area Under Curve (AUC) of the Receiver Operating Characteristic curve (ROC-curve), (Fawcett, 2006). In this performance measure, the binary classification threshold is moved while at each threshold value the True Positive Rate (TPR) and False Positive Rate (FPR) are noted. Plotting the TPR

[8] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

[9] `https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html`

versus the FPR results in the ROC-curve. Figure 4.6 shows the ROC-curve for two models, with the orange model outperforming the grey model. The area under this curve is the AUC, with the perfect model having an AUC of 1.

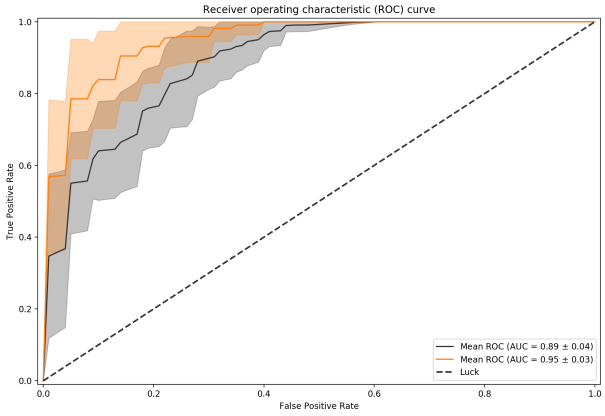

Figure 4.6: Averaged ROC-curves with standard deviations. The orange model is outperforming the grey model.

4.6.2 Statistical tests

When assessing the relative performance of two sets of features, two tests are used. As discussed in Section 2.4, computational time considerations can cause one to be preferable over another. However, in this research the emphasis is on confidently assessing the relative performance, so multiple tests are used.

5x2-cv with t-test

The 5x2-cv method, as discussed in 2.4.2, is used and for each set of features the average AUC and standard deviation are calculated. An independent two-sample t-test is performed with these statistics from two different models, to see if their performances are significantly different. The t-test is given by

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_p \sqrt{\frac{2}{a}}}, \tag{4.4}$$

where

$$s_p = \sqrt{\frac{s_{X_1}^2 + s_{X_2}^2}{2}}, \tag{4.5}$$

and where X_1 is the set of AUCs from the first model, X_2 is the set of AUCs from the second model and $a = 10$. t then follows Student's t-distribution with 18 degrees of freedom.

An important comment on the assumption of the independence of these two samples is that, in the case of this research, this can not be completely guaranteed. For example, when calculating the AUCs for the Basic and All features sets, the Basic set will be selected from a subset of the All set. In this way there can be overlap between the features that generate the AUCs of the different models. Literature on this specific problem could not be found, though somewhat similar problems did not excessively influence the results, (Nadeau & Bengio, 2003).

McNemar's test

McNemar's test uses a contingency table, given in Table 4.1, comparing the differently classified observations by two models. In the table, b is the number of observations that are correctly identified by model 1 and incorrectly identified by model 2 and for c the situation is reversed. McNemar's test statistic is $(b-c)^2/(b+c) \sim \chi^2_1$. It should be noted that in the case of $(b+c) < 25$, the exact binomial test or $(|b-c|-1)^2/(b+c)$ should be used, as suggested by Edwards (1948).

Table 4.1: Contingency table for McNemar's test.

	Model 2 correct	**Model 2 incorrect**	**Row total**
Model 1 correct	a	b	a + b
Model 1 incorrect	c	d	c + d
Column total	a + c	b + d	N

4.6.3 Out of time holdout set

When predicting with time series data, it is recommended to use an Out Of Time (OOT) holdout sample (Bergmeir & Benítez, 2012). This is a selection of the data which occurs after the training data. This is not possible on the public data set, due to all label assessments occurring at the same date. However, for the private data set this is possible and an OOT holdout sample is reserved. On this sample the final performance of the model will be measured. By keeping a holdout sample away from the modelling, the final model will not be biased towards it.

4.7 Benchmarking

To assess the scale-ability for the practical implementation of this feature generation approach, the feature generation and model training times are measured. The model training time is defined as the time taken by the optimal model from the grid search to train on all observations using the selected features. For the feature generation, the total computational time consists of several parts. First the data is preprocessed and split into time windows, as explained in Sections 3.2 and 4.1.1 respectively. Then the actual feature generation occurs. Finally, all features are combined into a large data frame and returned. For the benchmarking of the feature generation time, the duration of this entire process is noted for each features subset, as described in Section 4.4.1.

Chapter 5

Results

In this section, the results from the research are presented. First, the feature generation, mother wavelet, feature selection, model comparison and benchmarking are discussed. After this, a practical contribution in the form of an open source package is presented. Finally, the section concludes with a discussion.

5.1 Research results

In this section the results from the research are presented. In-depth results from the analysis on the public data is given, while only abstract results from the private data are given, in order not to leak confidential information.

5.1.1 Feature generation

Applying the methods proposed in Sections 3.3 and 4.1, the private data set consists of 1,408 accounts (704 defaults) with a total of 1,553,054 transactions. The public data set consists of 90 accounts (45 defaults) with a total of 7,941 transactions. For each account the following number and types of features, as explained in Section 4.2, are generated from both the balance and transaction data, not yet considering any transformations:

- Quarterly features over 4 quarters with per quarter:
  - 7, Basic
  - 88, Fourier complete
  - 20, Fourier n-largest, n = 10
  - 98, Wavelet complete, depth = 4
  - 56, Wavelet basic, depth = 4
- Yearly features over 1 year with per year:
  - 7, Basic
  - 380, Fourier complete
  - 60, Fourier n-largest, n = 30
  - 364, Wavelet complete, depth = 6
  - 84, Wavelet basic, depth = 6

In total there are 3,942 regular features per account, of which 70 Basic, 1,744 Fourier and 2,128 Wavelet features. On top of this, all features are also generated from the normalized data set, adding another 3,942 features. Finally, all features are also generated from both vectors of the PCA and ICA transformed data sets, adding 3,942 features for each transformation. In total this comes to 15,768 features.

5.1.2 Mother wavelet

In this section the influence of the mother wavelet is investigated. The following mother wavelets are used: 'db2', 'db4', 'coif2', 'sym3'. These wavelets are chosen as they were either used in previous feature generation research within the financial domain as discussed in Section 2.2.1, or are the more well known discrete mother wavelets. The shapes of these wavelets are given in the top row of Figure 5.1. Sets of regular features are generated with each mother wavelet for each data set. Then, for each set of features a grid search, as discussed in Section 4.5.2, is performed for the LightGBM model. Finally, for each optimal model, 5x2-cv, as discussed in Section 4.6, is used to assess the performances, which are given in Table 5.1.

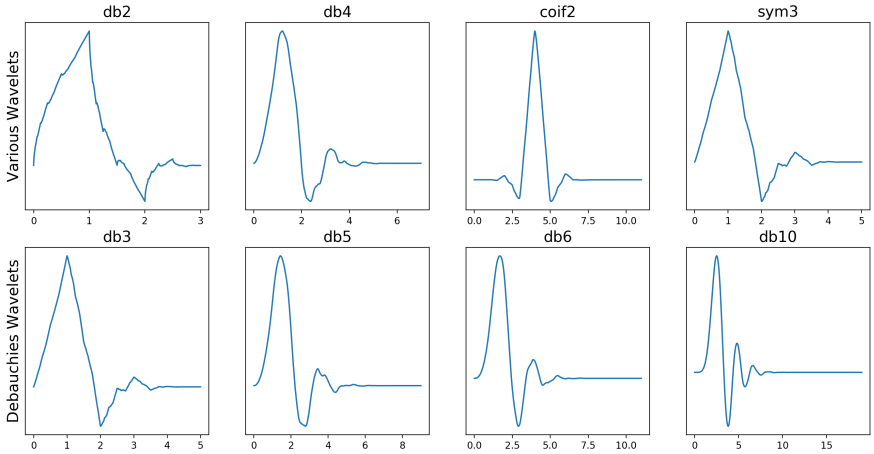

Figure 5.1: The used types of mother wavelets, generated with the PyWavelets package.

Table 5.1: Mother wavelet comparison: various.

Mother wavelet	**AUC (st. dev.) on private data**	**AUC (st. dev.) on public data**
db2	0.74 (0.011)	0.87 (0.029)
db4	0.73 (0.013)	0.87 (0.029)
coif2	0.72 (0.010)	0.87 (0.029)
sym3	0.72 (0.011)	0.87 (0.029)

In the public data set, there is no difference in performance. In the private data set it is seen that the Daubechies (db) wavelets achieve superior results, though mostly not statistically significant. However, for this reason 'db3', 'db5', 'db6' and 'db10' are tried as well. The shapes of these wavelets are given in the bottom row of Figure 5.1. Results are given in Table 5.2, where it is seen that 'db2' still performs equally well or better for the public and private data set, respectively. Because of this result, 'db2' is used for the rest of the research.

5.1.3 Feature selection

As discussed in Section 4.4, feature selection is performed on the three sets of features. The sets being only the regular basic features (Basic set), only the regular features (Regular set) and all features (All set). As seen in Figure 5.2, the LightGBM algorithm clearly profits from a reduced set of features, removing noise from the data. Both algorithms show a similar trend with each set of features. First, performance increases when going from a large number of features to a

Table 5.2: Mother wavelet comparison: Daubechies wavelets.

Mother wavelet	**AUC (st. dev.) on private data**	**AUC (st. dev.) on public data**
db2	0.74 (0.011)	0.87 (0.029)
db3	0.73 (0.011)	0.87 (0.029)
db4	0.73 (0.013)	0.87 (0.029)
db5	0.73 (0.011)	0.87 (0.029)
db6	0.73 (0.010)	0.87 (0.029)
db10	0.71 (0.015)	0.87 (0.029)

smaller number. Then, as seen in Figure 5.3, performance plateaus, before it finally decreases again. The number of features just before performance starts to drop, in this case 11, is optimal.

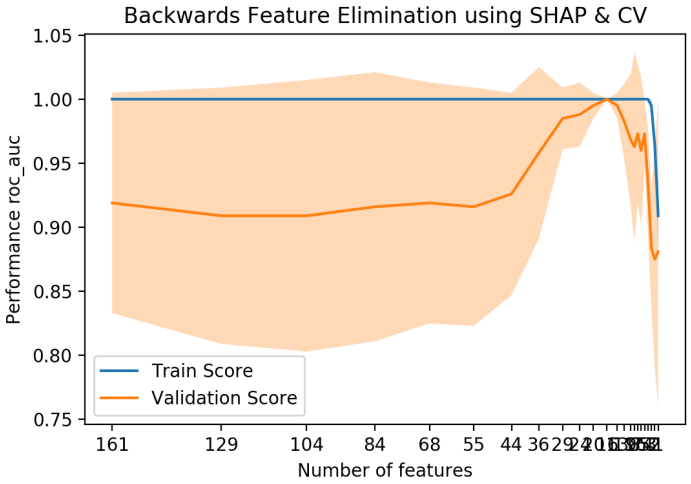

Figure 5.2: Recursive feature elimination with LightGBM for the Regular set from the public data.

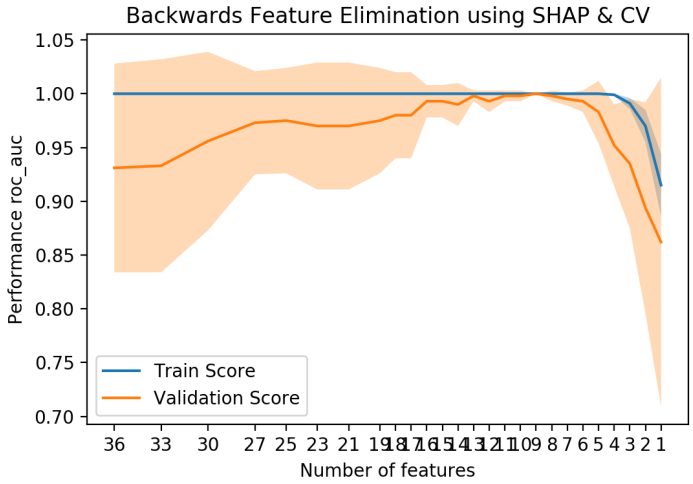

Figure 5.3: Final stage of recursive feature elimination with LightGBM for the Regular set from the public data.

This process is applied to both data sets with both algorithms and an overview of the optimal number of features for both data sets and algorithms is given in Table 5.3.

An overview of the final optimal selection of features for the public data set with LightGBM is given in Appendix A.2.

Table 5.3: Optimal number of features for the private and public data sets.

	Private data		**Public data**	
	LightGBM	Random Forest	LightGBM	Random Forest
Basic set	6	8	8	10
Regular set	33	19	9	10
All set	37	14	10	16

5.1.4 Model comparison

In this section, the performances of the optimal selected features from the Basic set, the Regular set and the All set are compared. Figure 5.4 shows the results, where 5x2-cv is used to compute the average AUC and standard deviation. A full overview of the statistical performance comparisons is given in Table 5.4.

Because the optimal number of features varies considerably in some cases, the best 15 features of each features set are also compared. The results are given in Figure 5.5.

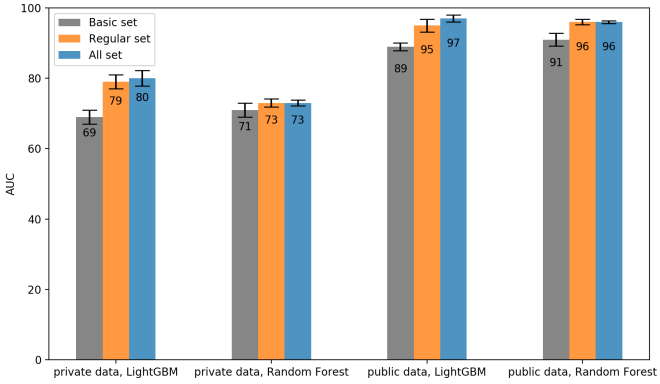

Figure 5.4: AUC comparison for the optimal number of features sets with 95% confidence interval.

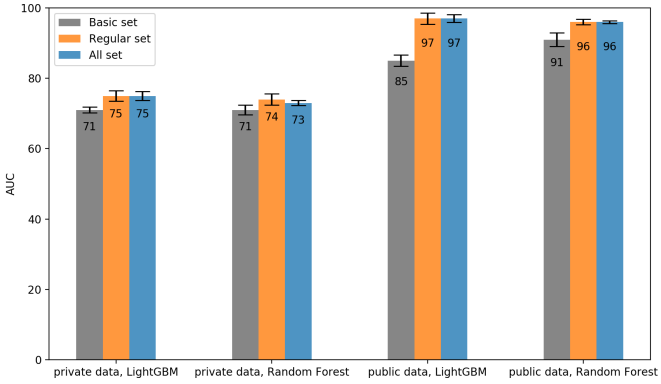

Figure 5.5: AUC comparison for the best 15 features from the feature sets with 95% confidence interval.

Table 5.4: Statistical results for optimal performance comparison of model a versus model b. * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

	Model		**Private data**		**Public data**	
	a	**b**	LightGBM	Random Forest	LightGBM	Random Forest
5x2-cv	Basic set	Regular set	b > a ***	b > a **	b > a **	b > a **
	Basic set	All set	b > a ***	b > a **	b > a **	b > a **
	Regular set	All set	a = b	a = b	a = b	a = b
McNemar's test	Basic set	Regular set	b > a **	b > a *	b > a *	a = b
	Basic set	All set	b > a **	b > a *	b > a *	a = b
	Regular set	All set	a = b	a = b	a = b	a = b

In Figures 5.6, 5.7 and 5.8, the respective averaged ROC curves of Basic set versus Regular set, Basic set versus All set and Regular set versus All set are given for the public data set with LightGBM.

As mentioned in Section 4.6, the private data set has an Out Of Time holdout sample. The performance on this holdout set is given in Table 5.5, together with the previously given cross-validated performance on the regular validation set.

5.1.5 Benchmarking

As explained in Section 4.7, by benchmarking the feature generation and the model training times, a measure of the scale-ability is observed. Table 5.6 shows this information for both data sets and algorithms.

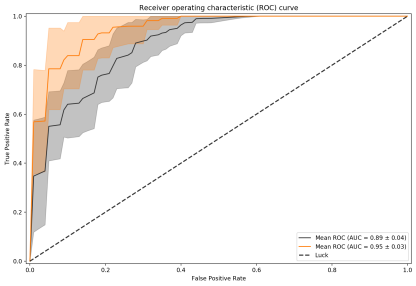

Figure 5.6: Averaged AUCs for the optimal Basic set versus Regular set with 95% confidence interval.

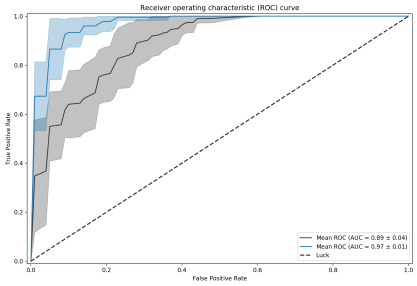

Figure 5.7: Averaged AUCs for the optimal Basic set versus All set with 95% confidence interval.

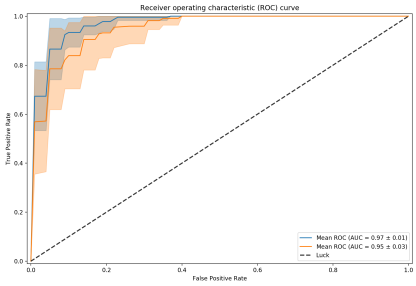

Figure 5.8: Averaged AUCs for the optimal Regular set versus All set with 95% confidence interval.

Table 5.5: Performance comparison of the Out Of Time holdout set with the validation set for the optimal number of features from the private data.

	LightGBM		**Random Forest**	
	OOT	Validation	OOT	Validation
Basic set	0.67	0.69 (0.020)	0.67	0.71 (0.019)
Regular set	0.75	0.79 (0.022)	0.70	0.73 (0.012)
All set	0.77	0.80 (0.020)	0.71	0.73 (0.008)

Table 5.6: Time taken to generate the sets of features and train the respective models.

Feature generation	**Private data**		**Public data**	
Basic set	1210 s		34 s	
Regular set	2387 s		66 s	
All set	9566 s		270 s	
Model training	**Private data**		**Public data**	
	LightGBM	Random Forest	LightGBM	Random Forest
Basic set	0.024 s	8 s	0.008 s	6 s
Regular set	0.030 s	8 s	0.011 s	6 s
All set	0.025 s	8 s	0.008 s	6 s

5.2 Open-source package

In an endeavor to contribute through a practical implementation of this research, the entire analysis is made publicly available on GitHub[1]. Instructions on how to acquire the public data set are included, together with functions for the preprocessing. On top of this, the feature generation from transaction data part is also programmed as an open-source package[2], with easy to use functions and complementary documentation[3].

5.3 Discussion

In this section the results are discussed and interpreted. Starting off with the performances of the different mother wavelets, it is seen that for the public data set there is no difference in performance. This is likely because the algorithm already achieves a relatively good performance

[1] `https://github.com/JanBargeman/Thesis-FE-SP`
[2] `https://pypi.org/project/SPOEF/`, install via: pip install SPOEF
[3] `https://janbargeman.github.io/SPOEF/index.html`

on a small number of observations, so the slight changes in mother wavelet shapes have no influence. For the private data set the Daubechies wavelets generally perform better. Though not statistically significant, within this family of mother wavelets, the 'db2' wavelet does get the best average result. This could be due to it being the least detailed wavelet in this family, meaning it only has to resemble the shape of the data loosely.

Then, focusing on the feature selection, it is seen that the All set only uses less features than the Regular set in the case of the private data set with Random Forest. For each other case, the All set uses more features than the Regular set, indicating that specifically the PCA and ICA transformations are not condensing the information as was expected. Besides this, the All set does generally outperform the Regular set, although not significantly. It is interesting to note that, as seen in Appendix A.2 in Table A.2, the Regular set contains one of the features from the Basic set, even though the All set does not. This minimum of the balances feature is likely replaced by a Wavelet representation of the minimum of the balance.

Looking at the results from the model comparisons, it is clearly seen that the signal processing features significantly increase the predictive performance. The LightGBM algorithm also profits considerably more from the increased features sets than the Random Forest. Furthermore, looking at Table 5.4, 5x2-cv is seen to have more power than McNemar's test, as was expected. On top of that, the increase in computational cost for this more powerful test was hardly noticed and therefore easily justified. Even more, McNemar's test is not even able to make claims for the public data set with Random Forest. This is likely because it only has 90 observations and the model already has a high AUC. Inspecting the differences in correct model predictions it turns out that $b + c$ is occasionally even less than 2, which is definitely too small for McNemar's test. For the Out Of Time holdout set from the private data it is seen that the performance drops slightly. This is not unexpected and the size of the drop is not problematic. The change in predictive performance is similar for all features sets and both algorithms.

When inspecting the averaged ROC-curves for the different features sets, it becomes clear that the improvement from the signal processing features is in both increasing the True Positive Rate as well as decreasing the False Positive Rate. This indicates that these features improve the model's ability to more successfully identify both defaulters and non-defaulters. On top of this, the All set does have a noticeable smaller confidence region than the Regular set, indicating that the features from the transformed data are stable predictors.

Finally, considering the benchmarked feature generation and model training times, it should be noted that feature generation, specifically for the private data, takes considerably longer. This is due to the transformations of the data and the Fourier and Wavelet Transforms. For even one feature from a Fourier or Wavelet Transform, the entire transform has to be calculated. Though it is concluded that with increased computing capacity and correct planning, the increase in performance is surely worth it. Finally, the model training times experience hardly any change, and stay relatively small.

Chapter 6

Conclusion

This research aimed to answer the research question of how signal processing features perform in comparison to the methodology of a major Dutch bank in predicting company default. This was done using transaction data from a private and public source. Fourier Transform and Wavelet Transform features were generated from the regular data and from normalized, PCA and ICA transformations of the data. Different mother wavelets were tried for the Wavelet Transform, where the Daubechies 2 wavelet had the best result and was used for the rest of the analysis. After this, the features were divided into three sets: Basic set, Regular set and All set. Then, feature selection was performed on each set using recursive feature elimination with the measure of feature importance being their Shapley value. The performances of the optimal number of features of each set were then calculated with the LightGBM and Random Forest algorithm, and compared with 5x2-cv followed by a t-test and McNemar's test.

This analysis indicated that the signal processing features contributed significantly to predicting company default for both data sets and both algorithms. For the private data set the performance improves from 0.69 AUC to 0.80 AUC with LightGBM and from 0.71 AUC to 0.73 AUC with Random Forest. For the public data set the performance improves from 0.91 AUC to 0.98 AUC with LightGBM and from 0.89 AUC to 0.94 AUC with Random Forest. The features from PCA, ICA and normalized versions of the data sets did not significantly improve the performance for either data set or algorithm, but did lower the standard deviation of the performance. The out of time performance of the private data was tested, showing an acceptable performance drop of 2% to 4% AUC, which was similar among features sets and algorithms. Finally, when looking at the ROC-curves it becomes clear that these features improve the model's ability to more successfully identify both defaulters and non-defaulters.

Future research

In banking it is very important that features are explainable and fair. Specifically the wavelet features are difficult to explain and have an unclear business interpretation. On top of this, the fairness of these features should be investigated by testing the models for robustness on various aspects. For example, clients requesting their loans or paying salaries in different months or days of the month. Robustness in general should also be investigated more thoroughly. For example, signal processing methods behave strangely for clients with very few transactions.

Another extension could be to attempt to reduce the noisiness from the data, mentioned in Section 2.2, and use signal processing filters. Besides filters to make the data more smooth, it would also assist in handling resonance. As seen for the Fourier Transform in Figure 4.2, there is strong resonance from the monthly signal, which disguises possible 2-monthly or quarterly signals.

Finally, signal processing methods rely heavily on the sampling frequency. The smaller the sampling interval, the more precise these methods become. For this research the data was aggregated onto the daily level. It could be interesting to increase this to a 6-hour period or even hourly, depending on the detailedness of the data. However, having to fill in a major part of the data will likely have severe consequences as well, which should be kept in mind.

References

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, *23*(4), 589–609.

Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, *26*(10), 1340–1347.

Ayata, D., Yaslan, Y., & Kamaşak, M. (2016). Emotion recognition via random forest and galvanic skin response: Comparison of time based feature sets, window sizes and wavelet approaches. In *2016 medical technologies national congress (tiptekno)* (pp. 1–4).

Barboza, F., Kimura, H., & Altman, E. (2017). Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, *83*, 405–417.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213.

Bilik, I., Longman, O., Villeval, S., & Tabrikian, J. (2019). The rise of radar for autonomous vehicles: Signal processing solutions and future research directions. *IEEE Signal Processing Magazine*, *36*(5), 20–31.

Bouckaert, R. R. (2003). Choosing between two learning algorithms based on calibrated tests. In *Icml* (Vol. 3, pp. 51–58).

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Cieslak, D. A., & Chawla, N. V. (2008). Learning decision trees for unbalanced data. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 241–256).

Cortes, C., & Vapnik, V. (1995). Support vector machine. *Machine Learning*, *20*(3), 273–297.

Daubechies, I., & Sweldens, W. (1998). Factoring wavelet transforms into lifting steps. *Journal of Fourier Analysis and Applications*, *4*(3), 247–269.

De Chazel, P., & Reilly, R. (2000). A comparison of the ecg classification performance of different feature sets. In *Computers in cardiology 2000. vol. 27 (cat. 00ch37163)* (pp. 327–330).

Dell'Agnola, F., Momeni, N., Arza, A., & Atienza, D. (2020). Cognitive workload monitoring in virtual reality based rescue missions with drones. In *International conference on human-computer interaction* (pp. 397–409).

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1923.

Dorfman, R. (1979). A formula for the gini coefficient. *The Review of Economics and Statistics*, 146–149.

Draper, B. A., Baek, K., Bartlett, M. S., & Beveridge, J. R. (2003). Recognizing faces with pca and ica. *Computer Vision and Image Understanding*, *91*(1-2), 115–137.

Du, B., Fernandez-Reyes, D., & Barucca, P. (2020). Image processing tools for financial time series classification. *arXiv preprint arXiv:2008.06042*.

Edwards, A. L. (1948). Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. *Psychometrika*, *13*(3), 185–187.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, *27*(8), 861–874.

García, V., Marqués, A. I., & Sánchez, J. S. (2015). An insight into the experimental design for credit risk and corporate bankruptcy prediction systems. *Journal of Intelligent Information Systems*, *44*(1), 159–189.

Giri, D., Acharya, U. R., Martis, R. J., Sree, S. V., Lim, T.-C., VI, T. A., & Suri, J. S. (2013). Automated diagnosis of coronary artery disease affected patients using lda, pca, ica and discrete wavelet transform. *Knowledge-Based Systems*, *37*, 274–282.

Gurney, K. (2018). *An introduction to neural networks*. CRC press.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*(1), 389–422.

Härdle, W., Lee, Y.-J., Schäfer, D., & Yeh, Y.-R. (2009). Variable selection and oversampling in the use of smooth support vector machines for predicting the default risk of companies. *Journal of Forecasting*, *28*(6), 512–534.

Hillegeist, S. A., Keating, E. K., Cram, D. P., & Lundstedt, K. G. (2004). Assessing the probability of bankruptcy. *Review of Accounting Studies*, *9*(1), 5–34.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proc. of the int'l conf. on artificial intelligence* (Vol. 56).

Jolliffe, I. (2005). Principal component analysis. *Encyclopedia of Statistics in Behavioral Science*.

Kahya, Y. P., Yeginer, M., & Bilgic, B. (2006). Classifying respiratory sounds with different feature sets. In *2006 international conference of the ieee engineering in medicine and biology society* (pp. 2856–2859).

Kang, S., Zhang, H., & Kang, Y. (2010). Application of signal processing and neural network for transient waveform recognition in power system. In *2010 chinese control and decision conference* (pp. 2481–2484).

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*, 3146–3154.

Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, *34*(11), 2767–2787.

Khemakhem, S., & Boujelbene, Y. (2018). Predicting credit risk on the basis of financial and non-financial variables and data mining. *Review of Accounting and Finance*.

Kruppa, J., Schwarz, A., Arminger, G., & Ziegler, A. (2013). Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications*, *40*(13), 5125–5131.

Lee, J. J., Lee, S. M., Kim, I. Y., Min, H. K., & Hong, S. H. (1999). Comparison between short time fourier and wavelet transform for feature extraction of heart sound. In *Proceedings of ieee. ieee region 10 conference. tencon 99.'multimedia technology for asia-pacific information infrastructure'(cat. no. 99ch37030)* (Vol. 2, pp. 1547–1550).

Liberati, C., & Camillo, F. (2018). Personal values and credit scoring: new insights in the financial prediction. *Journal of the Operational Research Society*, *69*(12), 1994–2005.

Ling, C. X., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *Kdd* (Vol. 98, pp. 73–79).

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 491–502.

Martis, R. J., Acharya, U. R., & Min, L. C. (2013). Ecg beat classification using pca, lda, ica and discrete wavelet transform. *Biomedical Signal Processing and Control*, *8*(5), 437–448.

Nadeau, C., & Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, *52*(3), 239–281.

Nanni, L., & Lumini, A. (2009). An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, *36*(2), 3028–3033.

Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. S. (2017). Learning feature engineering for classification. In *Ijcai* (pp. 2529–2535).

Nayak, M., & Panigrahi, B. S. (2011). Advanced signal processing techniques for feature extraction in data mining. *International Journal of Computer Applications*, *19*(9), 30–37.

Nyquist, H. (1932). Regeneration theory. *Bell System Technical Journal*, *11*(1), 126–147.

Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 109–131.

Prochazka, A., Kukal, J., & Vysata, O. (2008). Wavelet transform use for feature extraction and eeg signal segments classification. In *2008 3rd international symposium on communications, control and signal processing* (pp. 719–722).

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. *Nonlinear Estimation and Classification*, 149–171.

Shaker, M. M. (2006). Eeg waves classifier using wavelet transform and fourier transform. *Brain*, *2*(3).

Shapley, L. S. (1951). *Notes on the n-person game; ii: The value of an n-person game*. Santa Monica, CA: RAND Corporation. doi: 10.7249/RM0670

Stein, R. M. (2002). Benchmarking default prediction models: Pitfalls and remedies in model validation. *Moody's KMV, New York*, *20305*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Trustorff, J.-H., Konrad, P. M., & Leker, J. (2011). Credit risk prediction using support vector machines. *Review of Quantitative Finance and Accounting*, *36*(4), 565–581.

Willsky, A. S., & Young, I. T. (1997). *Signals and systems*. Prentice-Hall International.

Zhang, D., Ding, D., Li, J., & Liu, Q. (2015). Pca based extracting feature using fast fourier transform for facial expression recognition. In *Transactions on engineering technologies* (pp. 413–424). Springer.

Appendix A

A.1 Methodology: grid search

Table A.1: Hyperparameter grid for Random Forest and LightGBM.

	Parameter	Grid	Explanation
Random Forest	n_estimators	20, 30, 50, 100	Number of decision trees
	max_depth	3, 6, 9	Maximum depth of each tree
LightGBM	n_estimators	50, 100, 300	Number of boosted trees
	max_depth	3,6	Maximum depth of each tree
	reg_alpha	0, 20	Lasso regularization term on the weights
	reg_lambda	0, 50	Ridge regularization term on the weights

A.2 Results: final features

Table A.2: Alphabetical overview of features for the public data set with LightGBM when the optimal number of features is used.

Basic set	**Regular set**	**All set**
reg ba Q_1/4 B max	reg ba Q_1/4 wavelet depth_22	ICA 0 Y_1/1 wavelet depth_218
reg ba Q_2/4 B skew	reg ba Q_4/4 B min	ICA 1 Q_3/4 f2 real_27/44.5
reg ba Q_2/4 B std	reg ba Y_1/1 f2 real_162/182.5	PCA 0 Y_1/1 f2 imag_130/182.5
reg ba Q_4/4 B min	reg ba Y_1/1 fft size_11/30	norm tr Q_3/4 wav_B_detail depth_2 min
reg ba Y_1/1 B kurt	reg ba Y_1/1 wav_B_approx depth_1_min	reg ba Q_4/4 f2 imag_34/44.5
reg ba Y_1/1 B min	reg ba Y_1/1 wav_B_approx depth_2_min	reg ba Q_4/4 wav_B_approx depth_1_min
reg tr Q_2/4 B max	reg tr Q_2/4 f2 imag_33/44.5	reg ba Y_1/1 f2 real_7/182.5
reg tr Q_3/4 B sum	reg tr Y_1/1 fft index_8/30	reg ba Y_1/1 wav_B_approx depth_1_min
	reg tr Y_1/1 wavelet depth_55	reg ba Y_1/1 wav_B_approx depth_2_min
		reg ba Y_1/1 wav_B_approx depth_3_min

Table A.3: Alphabetical overview of features for the public data set with Random Forest when the optimal number of features is used.

Basic set	**Regular set**	**All set**
reg ba Q_1/4 B min	reg ba Q_4/4 f2 real_38/44.5	ICA 1 Q_1/4 f2 real_9/44.5
reg ba Q_2/4 B min	reg ba Q_4/4 wav_B_approx depth_4_min	ICA 2 Q_2/4 f2 real_9/44.5
reg ba Q_2/4 B std	reg ba Y_1/1 B min	PCA 0 Y_1/1 f2 real_7/182.5
reg ba Q_4/4 B mean	reg ba Y_1/1 f2 real_7/182.5	PCA 0 Y_1/1 fft size_10/30
reg ba Q_4/4 B min	reg ba Y_1/1 wav_B_approx depth_1_min	norm ba Y_1/1 f2 real_7/182.5
reg ba Q_4/4 B sum	reg ba Y_1/1 wav_B_approx depth_2_min	norm ba Y_1/1 wavelet depth_363
reg ba Y_1/1 B max	reg ba Y_1/1 wav_B_approx depth_3_min	reg ba Q_4/4 wav_B_approx depth_2_min
reg ba Y_1/1 B min	reg ba Y_1/1 wav_B_approx depth_4_min	reg ba Q_4/4 wav_B_approx depth_3_min
reg tr Q_2/4 B max	reg tr Q_1/4 f2 imag_34/44.5	reg ba Q_4/4 wav_B_approx depth_4_min
reg tr Q_2/4 B mean	reg tr Y_1/1 f2 imag_7/182.5	reg ba Y_1/1 B min
		reg ba Y_1/1 wav_B_approx depth_1_min
		reg ba Y_1/1 wav_B_approx depth_2_min
		reg ba Y_1/1 wav_B_approx depth_3_min
		reg ba Y_1/1 wav_B_approx depth_4_min
		reg tr Q_2/4 f2 imag_33/44.5
		reg tr Y_1/1 f2 imag_7/182.5