Machine Learning (2024) 113:3917–3928
https://doi.org/10.1007/s10994-021-06042-2

Special issue on feature engineering editorial

Tim Verdonck[1] · Bart Baesens[2] · María Óskarsdóttir[3] · Seppe vanden Broucke[4]

Published online: 6 August 2021

Abstract
In order to improve the performance of any machine learning model, it is important to focus more on the data itself instead of continuously developing new algorithms. This is exactly the aim of feature engineering. It can be defined as the clever engineering of data hereby exploiting the intrinsic bias of the machine learning technique to our benefit, ideally both in terms of accuracy and interpretability at the same time. Often times it will be applied in combination with simple machine learning techniques such as regression models or decision trees to boost their performance (whilst maintaining the interpretability property which is so often needed in analytical modeling) but it may also improve complex techniques such as XGBoost and neural networks. Feature engineering aims at designing smart features in one of two possible ways: either by adjusting existing features using various transformations or by extracting or creating new meaningful features (a process often called "featurization") from different sources (e.g., transactional data, network data, time series data, text data, etc.).

Keywords Featurization · Applied machine learning · Interpretability · Data engineering

1 Introduction

The main objective of machine learning is to extract patterns to turn data into knowledge for decision making. Since the beginning of this century, technological advances have drastically changed the size of data sets as well as the speed with which these must be analyzed. Modern data sets may have a huge number of instances, a very large number of features, or both. In most applications, data sets are compiled by combining data from different sources and databases (containing both structured and unstructured data) where each source of information has its strengths and weaknesses.

✉ Tim Verdonck
tim.verdonck@uantwerp.be

[1] Department of Mathematics, University of Antwerp, Antwerp, Belgium

[2] Faculty of Business and Economics, KU, Leuven, Belgium

[3] Department of Computer Science, Reykjavik University, Reykjavik, Iceland

[4] Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium

Before applying any machine learning algorithm, it is therefore necessary to transform these raw data sources into applicable and meaningful features that represent a certain property of the observations. Note that the features typically appear as columns in the data matrix that is given to the machine learning algorithm. This essential step, which is often referred to as *feature engineering* is of utmost importance in the machine learning process. The aim of feature engineering is twofold:

- Engineering (or *preparing*) the input data into features that the machine learning algorithm can understand and hence fulfill its requirements.
- Engineering (or *transforming*) variables into features to help the machine learning algorithms achieve better performance in terms of either predictive performance, interpretability or both.

The success of machine learning often depends strongly on the success of feature engineering. This is confirmed by Kaggle master Luca Massaron: *The features you use influence more than everything else the result. No algorithm alone, to my knowledge, can supplement the information gain given by correct feature engineering.* Also Andrew Ng, founder of `deeplearning.ai`, has stated that *Applied machine learning is basically feature engineering.*

It is clear from the definition that the feature engineering step may encompass many different methodologies to reach its goals. Some feature engineering techniques rely mainly on domain knowledge, others on intuition or are obtained through data analysis. Depending on the input data and available features, other techniques might be more successful. The gained increase in performance due to feature engineering is also sensitive to the choice of algorithm that is applied afterwards. There is no gold standard containing the best feature engineering techniques and therefore it is important to try out various approaches on your data and observe their effect on model performance. In general, the manual construction of features is often difficult and time-consuming.

2 Feature engineering: what and why

Feature engineering techniques are typically applied after gathering and cleaning the input data. In the cleaning step, one typically deals with missing values, errors, outliers and duplicates. Note that some machine learners or data scientists categorize certain cleaning steps (e.g., advanced anomaly detection or imputation techniques) as feature engineering as well. There is also a difference between feature selection, feature extraction and feature engineering, although they are sometimes used interchangeably. In the feature selection step, redundant or unused features are removed and as a result a subset of the original features is obtained. This can be done either before building the machine learning model or as part of the model building itself. Feature extraction reduces the dimension of the data set by creating new features. In principal component analysis for example, the new features are linear combinations of the original ones. Such dimension reduction techniques (which transform the original variables) are often also applied or investigated during the feature engineering step. Many feature engineering techniques exist and it is not always clear which techniques fall under the definition of feature engineering and which not.

3 Types of feature engineering

In what follows, we will give an overview of popular and often successful feature engineering techniques, without claiming that this list is complete. In this overview, we will also refer to the various articles that appear in this special issue.

We will first discuss univariate and multivariate feature engineering techniques for structured data. In these sections we focus on popular transformations and dimension reduction techniques. Note that structured data is data that can be easily represented in a tabular format with the rows typically representing the observations and the columns the variables or features. Then we will discuss the construction of features based on domain knowledge and some feature engineering techniques for time series data. We end this section by describing feature engineering for unstructured data, such as network, text, multimedia and geospatial data.

3.1 Univariate feature engineering

Univariate feature engineering for continuous variables can be handy for improving symmetry, normality or model fit. In credit risk modeling for example, a logarithmic transformation is often applied to size variables like loan amounts, asset sizes and gross domestic product (GDP) since these variables are typically non-negative and right skewed (Van Gestel et al 2022). The Box–Cox transformation is a well-known family of power transformation:

$$x \mapsto f(x;\lambda) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log(x) & \lambda = 0. \end{cases}$$

If $\lambda < 1$ then large values of X are compressed, whereas small values are more spread out and if $lambda < 1$, the inverse takes place. The first property makes the power transformation interesting if the distribution of X is skewed to the right, the second if X is skewed to the left (Which occurs less in practice). The transformation $f(x;\lambda)$ is monotone increasing and differentiable with respect to x and is monotone and continuous with respect to the parameter λ. Box–Cox can only be applied on strictly positive numbers. If this is not the case, the data can first be shifted (adding a positive constant that makes all observations positive). Alternatively, one can use the Yeo–Johnson power transformations, which are defined as:

$$x \mapsto f(x;\lambda) = \begin{cases} ((1+x)^\lambda - 1)/\lambda, & \lambda \neq 0,\ x \geq 0, \\ \log(x+1), & \lambda = 0,\ x \geq 0, \\ -((1-x)^{2-\lambda} - 1)/(2-\lambda), & \lambda \neq 2,\ x < 0, \\ -\log(-x+1), & \lambda = 2,\ x < 0. \end{cases}$$

The λ parameter of the Box–Cox and Yeo–Johnson transformation is typically obtained using maximum likelihood, but can also be set based on expert input or using experimentation (e.g., by optimizing the AUC).

Note that the maximum likelihood estimator is highly sensitive to outliers. Therefore, Raymaekers and Rousseeuw (2021) propose in this special issue a modification of both transformations as well as an estimator of the λ parameter that is robust to outliers.

Popular examples of univariate feature engineering for categorical data are dummy coding, percentile coding, thermometer coding for ordinal categorical variables (e.g., credit ratings), weights of evidence (WOE) coding, Owen Zhang style leave-one-out encoding or hashing (the “hashing trick”) (Baesens et al 2016).

In recent years, practitioners and scholars have also been adopting the usage of representational learning techniques in order to transform categorical variables (Guo and Berkhahn 2016). Here, levels of a categorical variable are mapped into a lower-dimensional (Euclidean) space, which describe the “entity embeddings” of the categorical variable and are sparser than the representation which would be obtained by simple one-hot dummy coding. The mapping itself is typically learned by a neural network. The label can be used as a supervised target, but unsupervised approaches can be used as well (t-SNE, discussed below, for instance, can be regarded as an unsupervised approach to construct an embedding as well). For a deeper discussion on this, we refer to Hancock’s recent survey (Hancock and Khoshgoftaar 2020).

3.1.1 Multivariate feature engineering

Principal component analysis (PCA) is a multivariate feature engineering technique that is aimed at reducing the dimensionality of the data by creating new features that are linear combinations of the original variables. The idea of PCA is to summarize the information in the large set of original variables by a small number of new features called principal components (PCs), which maintain, as much as possible, the variance in the original data. The PCs themselves are uncorrelated and ordered such that the first few retain most of the variation present in all of the original variables.

Essentially, PCA is a kind of matrix factorisation method where the idea is to factorize or decompose a data matrix into a product of other matrices which provide a more condensed, focused representation of the information in the data.

Besides PCA, other popular examples of matrix factorization methods are singular value decomposition (SVD), (non-negative) UV decomposition and tensor decomposition. Yet another example is Linear Discriminant Analysis (LDA) which tries to find an optimal linear transformation of the data such that each high dimensional observation is projected into a low dimension representation, while maintaining the original cluster structure and achieving maximum class separability. The LDA method often boils down to solving a trace-ratio problem to maximize the between-class scatter distance while minimizing the within-class scatter distance. It was long assumed that this problem has no closed-form solution, and one has to use time consuming iterative algorithms. In their paper, Shi and Wu (2021) propose a closed-form solution for the trace-ratio problem, and introduce two algorithms to solve it.

t-SNE stands for Distributed Stochastic Neighbor Embedding and was developed by Van der Maaten and Hinton (2008). Essentially t-SNE is also dimensionality reduction technique, comparable to PCA. Hence, it could also be considered as a multivariate feature engineering alternative. However, PCA is a linear dimension reduction technique that seeks to maximize variance and preserves large pairwise distances. In other words, things that are different end up far apart. This can lead to suboptimal reduction with non-linear data. On the contrary, t-SNE seeks to preserve local similarities by focusing on small pairwise distances. More specifically, t-SNE is a non-linear dimensionality reduction technique based on manifold learning. It assumes that the data points lie on an embedded non-linear manifold within a higher-dimensional space. A manifold is a topological space that locally resembles a Euclidean space near each data point. Examples are a 2 dimensional manifold, locally resembling a Euclidean plane near each point, or a 3D surface which can be described by a collection of such 2 dimensional manifolds. A higher dimensional space can thus be well "embedded" in a lower dimensional space. Other manifold reduction techniques are multidimensional scaling, isomap, locally linear embedding, auto-encoders and UMAP (Van Der Maaten et al 2009). Especially the latter has become a popular replacement of t-SNE in recent years.

In this special issue, Sürer et al (2021) present coefficient tree regression (CTR), which discovers the group structure, engineers derived features accordingly and fits the resulting regression model.

For removing redundant and irrelevant features in high-dimensional data, Hancer (2021) present in this special issue an improved evolutionary wrapper–filter approach which integrates an initialisation scheme and a local search module based on fuzzy mutual estimator. The proposed approach yields a lower computational cost and better classification performance.

3.2 Domain specific feature engineering

Business problems in different industries typically require different features. Therefore, domain or expert knowledge of the business problem is often important to create smart features that boost the performance of the machine learning algorithm. Based on expert knowledge one can for example add indicator features for a certain condition or construct a new interaction feature by taking combinations of two or more existing ones.

As a concrete example, consider the problem of churn prediction for prepaid contracts in Telco. Business experts know that a peak usage before the expiry date is an early warning that a consumer is consuming all his/her remaining credit and as thus an early warning for upcoming churn behavior. An example of a red flag indicator that fraud investigators have found by working in the field is *severe car accident but no doctor present at the accident scene* or *two insurance claims for severe surgery at the same day*. An interesting feature in human resources (HR) analytics is a binary indicator whether someone recently changed his/her LinkedIn profile as this could be an early sign that the employee is on the lookout for another job opportunity and hence at churn risk.

A popular example of domain specific feature engineering are the RFM features. RFM stands for Recency, Frequency and Monetary and has been popularized by Cullinan (1977) in a marketing setting. The RFM features are typically summarized from transactional data as follow: Recency measures the time since the most recent transaction. Frequency measures the total number of transactions in a given time span. And finally, monetary measures the value of transactions within the period examined. Basically, the RFM famework summarizes the transactions according to 3 dimensions. Each of these constructs can then be operationalized in various ways. For example, one can consider the minimum/maximum/average/most recent monetary value of transactions. Note that besides marketing analytics (Baesens et al 2002; Blattberg et al 2008), these features have also been successfully used in fraud analytics (Van Vlasselaer et al 2015; Baesens et al 2021).

Domain specific features can also be defined by considering combinations or interactions of variables. An example could be in real estate analytics, where a premium property feature can be defined based on the features number of bedrooms (e.g., bigger than 3) and number of bathrooms (e.g., larger than 2). At Kaggle competitions, it is often shown that combining certain pairs of variables or features yields more informative features than using them separately in the model.

3.3 Feature engineering for time series data

Time series data is available in many domains such as finance, engineering, health and economics. These data contain key information and are widely used for forecasting (i.e. predicting future events and extrapolating how a system evolves based on historical and current data). However, analyzing time series is not a trivial exercise. Some popular forecasting techniques are Autoregressive Integrated Moving Average (ARIMA), Vector Autoregression (VAR) and exponential smoothing. For more information, we refer to Hyndman and Athanasopoulos (2018). Most machine learning algorithms cannot be directly applied to time series data. A popular way to solve this issue is by transforming the time series into a feature-vector representation and then applying a traditional predictive model on this new representation. Setting this feature-vector representation is a crucial step to learn from time series data and has a major impact on the performance of the model. Constructing these features manually is a very challenging and time-consuming task. Note that this feature engineering step can also improve the forecasting techniques that were mentioned above.

In this special issue, Cerqueira et al (2021) present VEST (Vector of Statistics from Time series), which is a framework for automatically extracting an optimal representation from a univariate time series. This automatic feature engineering approach first transforms the input time series into several distinct representations, which are then summarized using statistics. After a feature selection process, the final set of features across the available representations is coupled with an autoregressive (AR) model.

Many real-world applications require analyzing several time series simultaneously. Such data are often collected using various sensors (e.g., measuring temperature and pressure at different locations in a company or measuring heart rate and blood oxygen of a person). In such settings, it is important that some features are obtained by combining values from different series. Manual feature engineering is even more intensive now and therefore De Brabandere et al (2021) propose in this special issue an automated feature construction system for multiple time series data, called TSFuse, which supports fusion and explores the search space in a computationally efficient way.

3.4 Feature engineering for network data

Similar techniques exist for networked data. Traditionally, instances in datasets are treated as independent entities. However, in several instances it is appropriate to think of the instances as a network, having relationships or links between them. Networks are a general language for describing and analyzing entities with relations or interactions (Newman 2018; Barabási 2016). Examples of networks include social networks, communication networks, neural networks in our brains, biological networks and networks in economics. A prominent feature in networks is that nodes with similar properties are often linked with each other, which means that the structure of the network can be utilized when making inferences about the nodes. This has been successfully achieved to predict churn in the telecommunication industry using call networks (Óskarsdóttir et al 2017; Verbeke et al 2014), for fraud detection in social security and insurance (Van Vlasselaer et al 2017; Óskarsdóttir et al 2021) and for credit scoring (Óskarsdóttir et al 2019).

The network structure, the position of the nodes within it and the interdependencies between them can say a great deal about their properties. However, feature engineering must be applied to extract this information so that it can be used in machine learning models. The result of the featurization process are new features that describe some of the nodes' properties in relation to their neighbors and position in the network. There are two main approaches to featurize the network. This can either be done manually or automatically.

Manual feature engineering entails hand-crafting predefined features. Firstly, there are centrality measures that quantify the nodes' importance in the network (Barabási 2016). These include the degree and PageRank (Page et al 1999; Barabási 2016) centralities, and in addition, in networks that are not too big, betweenness and closeness. Secondly, there are features that describe the properties of connected nodes, so called link-based features (Getoor 2005). Assume that a telecommunication provider wants to predict which of their customers are most likely to churn, and in addition to basic customer variables they have the call network as well. One example of a link-based feature is how many of a customers connections have already churned. Having many such neighbors in the network might be an indication that the customer will also churn, since most of the people they call have already left. Another set of link-based features can be obtained by considering some statistic of the properties of neighbors, i.e. the mean or median age of the connected customers. If the network is weighted the weighted average of neighbors properties can be calculated. Thirdly, influence from nodes with a known label can be propagated thought the network to quantify the influence of the property, i.e. churn. Examples of such propagation algorithms, are spreading activation and personalized PageRank (Page et al 1999; Dasgupta et al 2008).

Networks can also be featurized automatically. The goal here is to start from a network and learn a mapping function which maps each network node to a d-dimensional vector of features, also known as an embedding or representation of the nodes in the network. Again, the latter can then be given to a traditional machine learning algorithm to perform classification, clustering, etc. The idea here is to maximize the likelihood of preserving network neighborhoods of nodes in the low d-dimensional space. Random walk approaches such as DeepWalk (Perozzi et al 2014) and node2vec (Grover and Leskovec 2016) are commonly used. More recently, there have been substantial advances in applications where deep learning is applied to data generated from non-Euclidean domains, represented as networks with complex relationships and interdependencies between objects. Graph Neural Networks (GNNs) are a class of deep learning methods designed to perform inference on data described by networks. These methods are capable of both generating embeddings to use in a downstream machine learning tasks and of learning directly from the network. GNNs are neural networks that can be directly applied to networks, and provide an easy way to do node-level, edge-level, and graph-level prediction tasks (Hamilton et al 2017; Zhou et al 2020).

3.5 Feature engineering for multimedia data

Another popular example of unstructured data is multimedia data. Multimedia data is actually defined as data representing multiple types of medium to capture information. Popular examples of multimedia data are audio, video and image data.

The idea of feature engineering for unstructured data is to extract featurs such that these can be fed into a classical machine learning technique (e.g., decision tree, neural network, XGBoost) for pattern recognition.

For image data, various featurization techniques exist, depending on the particular goal or task at hand. In the area of computer vision research, for example, "feature detection" describes the task of finding specific patterns (features) which are unique to a particular image and limited in number and size (so that they can be easily compared). Many algorithms to find such features have been devised over the years, with ORB (Rublee et al 2011) being one notable, free-to-use non patented example. Many other techniques can be applied as well to transform images to a more representative version. Fourier Transformation (or Spectral Analysis), is a well-known image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, which might offer an "easier" representation for machine learning techniques to learn from. Also very common is the application of filters for edge detection, blurring, etc. In more recent years, however, many of such hand-crafted approaches have been replaced by applying deep learning based techniques on image data sets, which apply a stack of convolutional filters which are learned during training to ultimately predict a target from. In a sense, such convolutional neural networks perform a form of automated feature engineering, though sadly in a form which is relatively black box. Still, over the past years, interpretability techniques have been devised in order to extract insights from the neural network. Also, it is quite common that the lower-level outputs of a convolutional network are used to create a representational vector of images—providing a sparser description which can then e.g., be utilized by a traditional machine learning technique. Finally, it is also worth mentioning that in many cases, the output of a deep learning model is used as a feature in a model which is more white box. As an example, consider a neural network which is trained to detect the presence of a swimming pool in satellite imagery, though where its output is used as a variable in another, simpler model to e.g., predict the price of a property. How the network detects a swimming pool is perhaps of lesser importance (given a good performance), though how the presence of a swimming pool impacts the price should be understandable and interpretable.

For audio data, we find that a similar setting exists here. Fourier Transformations are commonly applied here too, as well as frequency filters, beat and pitch detection techniques, and so on. Nevertheless, here too we see more and more a direct application of deep learning. Video then combines these two data formats.

3.6 Feature engineering for text data

Text is another form of unstructured data where understanding the context, i.e. of sentences and documents, is necessary in order to interpret their meaning. Text data typically contain much higher dimensions than traditional data and are used in many applications such as spam filtering, marketing, sentiment analysis and fraud detection. Featuring engineering is a key step in text mining, where numerical features are created from the raw text data. Different techniques exist to achieve this.

The first traditional approach is the Bag-of-Words model, which builds a vocabulary from a set of documents (corpus), and counts how often each word appears in each document. However, such term frequency can give a wrong representation of the text. The more advanced Term frequency–inverse document frequency (TF–IDF) can be used to quantify the importance of words relative to documents in a corpus, where the value of a word increases proportionally by count but is inversely proportional to the frequency of the word in the corpus (Rajaraman and Ullman 2011). This approach is commonly used in text-based recommender systems (Beel et al 2016).

Word embedding models are a popular family of methods capable of transforming raw text into real valued vectors that encode the meaning of the words or their context. As a result, words that are closer in the resulting vector space are usually similar in meaning

(Jurafsky and Martin 2000). They can be obtained by using language modeling and feature learning techniques. The first model of this family was word2vec which uses shallow neural network architectures to produce vector spaces of several hundreds of dimensions, where two words with similar context usually appear close to each other in the vector space (Mikolov et al 2013). This is achieved by either by starting from a single word to predict its context (Skip-gram) or starting from the context to predict a word (Continuous Bag-of-Words). The next word embedding model to appear was GloVe (Global Vectors for Word Representation) which builds word embeddings such that a combination of word vectors relates directly to the probability of these words' co-occurrence in the corpus thus stressing the importance carried by the information of the frequency of co-occurrences (Pennington et al 2014). Finally, fastText was developed to make up for the lack of generalizability of its predecessors (Bojanowski et al 2017). FastText is capable of generalizing to unknown words. In addition it needs less training data.

In the last years, state of the art language models which use transfer learning from attention-based transformers have completely revolutionized the NLP landscape. Popular models for such contextualized and dynamic word embeddings are ELMO (Peters et al 2018) and BERT (Devlin et al 2018).

Advanced machine learning models learning from text data are typically very hard to interpret. In the context of high-dimensional data, rule-extraction techniques typically leads to many rules without a comprehensible explanation. To solve this issue, Ramon et al (2021) propose a rule-extraction methodology based on higher-level, less-sparse *metafeatures*. They show that explanation rules extracted with data-driven metafeatures yield better results than those extracted using the textual features on which the model was trained.

Topic modeling is a text mining technique that extracts the most important topics and their accompanying keywords whereas sentiment analysis determines the emotion behind the words (whether it is positive or negative). In this special issue, Loginova et al (2021) investigate the usefulness of adding features extracted from textual data with sentiment analysis in forecasting directional bitcoin price returns. Their dataset includes various textual sources, such as financial data from CryptoCompare, search queries from Google Trends and textual data from forums, Reddit and news.

3.7 Feature engineering for geospatial data

Finally, geospatial data contain information about the location of objects or events (typically as coordinates on the earth). The location information is often also combined with temporal and weather information. The location may be static or dynamic and can be obtained from various sources (e.g., satellite imagery, cell phone data, sensor data and social media data). Large amounts of geospatial data are publicly available and this information is of interest in many applications, certainly when it can be added to traditional business data.

Geospatial data can be featurized using either a point, raster or vector format. A point format represents a geographic location in terms of its longitude and latitude. A raster format represents geographic data as a grid of values which are rendered on a map as pixels. Each pixel then corresponds to an area or cell on the Earth's surface. Finally, the vector format uses vectors which are composed of discrete geometric locations known as vertices that define the shape (e.g., point, line or polygon) of a spatial object.

Some insurance companies nowadays obtain continuously personalized car driving information (telematics data) about their policyholders. To include such data in actuarial predictive models, feature engineering is needed. In this special issue, Gao et al (2021) propose two neural networks to extract driver risk information from telematics data respresented by speed-acceleration heatmaps. The neural networks simultaneously perform feature engineering and regression modeling.

4 Conclusions

In this paper accompanying the special issue on Advances in Feature Engineering in Machine Learning we have given a comprehensive overview of feature engineering methods and practices in various fields of data science and machine learning, ranging from univariate and domain specific feature engineering, to advanced and sophisticated methods that are capable of transforming information hidden in complex and unstructured data, such as networks, text and locations, into representative and informative features, that can be used in down stream machine learning tasks. Furthermore, we have presented the state-of-the-art in feature engineering in various domains, such as for time series and telematics data, with the goal of achieving more accurate forecasting results and enhanced explainability, to name a few. As such, we believe it could prove useful for any student or practitioner who wants to dive into machine learning and learn about its vital feature engineering step.

The current machine learning landscape is covered with advanced and intricate deep learning architectures capable of learning from massive amounts of complex and unstructured data with a wide range of applications in more powerful ways than we could have ever imagined. The actual feature engineering process is more and more becoming an inherent part of the models' architecture, e.g., through convolutional layers. At the same time, the models are more black box than ever before, which can cause more harm than good, when care is not taken. Therefore we believe that significant effort should still be put into feature engineering, to ensure that machine learning models remain as explainable and interpretable as possible.

We would like to end by thanking Peter Flach and Hendrik Blockeel, the previous and current Editor-in-Chief of Machine learning, and Dragos D. Margineantu, the Action Editor for Special Issues, for giving us this opportunity, as well as the authors, reviewers, and associate editors, without whose valuable contributions, the special issue would not have been a reality.

References

Baesens, B., Höppner, S., & Verdonck, T. (2021). Data engineering for fraud detection. *Decision Support Systems*, 113492.

Baesens, B., Roesch, D., & Scheule, H. (2016). *Credit risk analytics: Measurement techniques, applications, and examples in SAS*. Hoboken: Wiley.

Baesens, B., Viaene, S., Van den Poel, D., Vanthienen, J., & Dedene, G. (2002). Bayesian neural network learning for repeat purchase modelling in direct marketing. *European Journal of Operational Research, 138*(1), 191–211.

Barabási, A. L. (2016). *Network science*. Cambridge: Cambridge University Press.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). paper recommender systems: A literature survey. *International Journal on Digital Libraries, 17*(4), 305–338.

Blattberg RC, Kim BD, & Neslin SA (2008). Why database marketing? In *Database marketing* (pp. 13–46). Berlin: Springer, New York

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5,* 135–146.

Cerqueira, V., Moniz, N., & Soares, C. (2021) Vest: Automatic feature engineering for forecasting. *Machine Learning*, 1–23.

Cullinan, G. J. (1977). Picking them by their batting averages' recency-frequency-monetary method of controlling circulation. Manual release 2103.

Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A. A., & Joshi, A. (2008). Social ties and their relevance to churn in mobile telecom networks. In *Proceedings of the 11th international conference on extending database technology: Advances in database technology* (pp. 668–677).

De Brabandere, A., Op De Beéck, T., Hendrickx, K., Meert, W., & Davis, J. (2021). Automating feature construction for multi-view time series data. *Machine Learning*, 1–40.

Devlin, J., Chang, M. W., Lee. K., & Toutanova, K. (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805.

Gao, G., Wang, H., & Wüthrich, M. V. (2021). Boosting Poisson regression models with telematics car driving data. *Machine Learning*, 1–30.

Getoor, L. (2005). Link-based classification. In U. Maulik, L. B. Holder, & D. J. Cook (Eds.), *Advanced methods for knowledge discovery from complex data* (pp. 189–207). Berlin: Springer.

Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp 855–864).

Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. *CoRR*, abs/1604.06737, arxiv:1604.06737.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 1025–1035).

Hancer, E. (2021). An improved evolutionary wrapper-filter feature selection approach with a new initialisation scheme. *Machine Learning*, 1–24.

Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of Big Data, 7,* 1–41.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. London: OTexts.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Hoboken: Prentice Hall.

Loginova, E., Tsang, W.K., van Heijningen, G., Kerkhove, L.P., & Benoit, D.F. (2021). Forecasting directional bitcoin price returns using aspect-based sentiment analysis on online communities data. Machine Learning pp 1–30.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781.

Newman, M. (2018). *Networks*. Oxford: Oxford University Press.

Óskarsdóttir, M., Ahmed, W., Antonio, K., Baesens, B., Dendievel, R., Donas, T., & Reynkens, T. (2021). Social network analytics for supervised fraud detection in insurance. *Risk Analysis* https://doi.org/10.1111/risa.13693.

Óskarsdóttir, M., Bravo, C., Sarraute, C., Vanthienen, J., & Baesens, B. (2019). The value of big data for credit scoring: Enhancing financial inclusion using mobile phone data and social network analytics. *Applied Soft Computing, 74,* 26–39.

Óskarsdóttir, M., Bravo, C., Verbeke, W., Sarraute, C., Baesens, B., & Vanthienen, J. (2017). Social network analytics for churn prediction in telco: Model building, evaluation and network architecture. *Expert Systems with Applications, 85,* 204–220.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford InfoLab.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:180205365.

Rajaraman, A., & Ullman, J. D. (2011). *Data mining* (pp. 1–17). Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9781139058452.002.

Ramon, Y., Martens, D., Evgeniou, T., & Praet, S. (2021). Can metafeatures help improve explanations of prediction models when using behavioral and textual data? *Machine Learning*, 1–40.

Raymaekers, J., & Rousseeuw, P. J. (2021). Transforming variables to central normality. *Machine Learning*, 1–23.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, IEEE (pp 2564–2571).

Shi, W., & Wu, G. (2021). New algorithms for trace-ratio problem with application to high-dimension and large-sample data dimensionality reduction. *Machine Learning*, 1–28.

Sürer, O., Apley, D.W., & Malthouse, E. C. (2021). Coefficient tree regression: Fast, accurate and interpretable predictive modeling. *Machine Learning*, 1–38.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(11), 2579–2605.

Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: A comparative. *Journal of Machine Learning Research, 10*(66–71), 13.

Van Gestel, T., Martens, D., & Baesens, B. (2022). *Predictive analytics: Techniques and applications in credit risk modelling*. Oxford: Oxford University Press.

Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2015). Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems, 75,* 38–48.

Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2017). Gotcha! Network-based fraud detection for social security fraud. *Management Science, 63*(9), 3090–3110.

Verbeke, W., Martens, D., & Baesens, B. (2014). Social network analysis for customer churn prediction. *Applied Soft Computing, 14,* 431–446.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., et al. (2020). Graph neural networks: A review of methods and applications. *AI Open, 1,* 57–81.