university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Preprocessing Pipeline Meta-Learning Using Reliable and Diverse Meta-Target Selection

Charlie Albietz

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

University of Groningen

Preprocessing Pipeline Meta-Learning, Using Reliable and Diverse Meta-Target Selection

Master's Thesis

To fulfil the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Dr. Herbert Jaeger (Artificial Intelligence, University of Groningen)
and
Maik Frye, MSc. (Fraunhofer Institute for Production Technology)

Charlie Albietz (s3058739)

April 6, 2023

Contents

Acknowledgments

This project has proven tougher than initially expected and I would like to thank all those who have supported me throughout the process. I would like to thank those working at Fraunhofer IPT for allowing me access to the data required to train the models used in this project and specifically Maik for the supervision. I would also like to thank all those who participated in reviewing and proof reading the many drafts of this thesis. I would especially like to thank Lynne for everything she did. This project is dedicated to my Grandad who would have been proud.

Abstract

Meta-learning, or learning about learning itself, can be applied to many branches of Machine Learning (ML). One of such branches aims, like Automated Machine Learning (AutoML) systems, to find ideal ML model parameters for novel datasets. Meta-learning systems obtain information about how model parameters will likely affect datasets by evaluating the performance of many previously run ML models. With this information, meta-models can be trained to predict the performance of future ML algorithms depending on dataset characteristics and model settings. In this paper, a meta-learning system was trained to predict the performance of dataset preprocessing (DPP) pipelines, allowing for automation of the DPP pipeline optimisation process. Previous meta-learning studies have highlighted a need for improvements concerning meta-model accuracy when predicting ideal DPP pipelines. For this reason, the current project aims to employ a new method that is expected to help increase meta-model accuracy for DPP, as well as producing strong DPP pipeline recommendations for incoming datasets. The method that is proposed in this project is to improve how meta-targets are selected. Meta-targets are targets that a meta-learner is trying to predict. In a scenario where DPP pipeline scores are to be predicted by the meta-learner, it is a common practice to define a smaller set of DPP pipelines to predict the performance of, rather than the entire set of possible DPP pipeline combinations. In this project, a novel method for choosing this subset of DPP pipelines is investigated. Usually, this subset consists of well-performing DPP pipelines chosen via the introduced principle of reliability: each DPP pipeline within the subset can show a high average performance across the training data. The top pipelines are then chosen according to a metric. An alternative method for meta-target selection is proposed: the principle of diversity. In this method, diverse meta-targets with high maximum performances for mutually exclusive groups of training inputs are chosen, allowing for specialist DPP pipelines to be considered in place of average performers. The current meta-learner was trained and tested on classification and regression datasets using reliable and diverse meta-target selection methods. The results show that the meta-learning accuracy increased when using diverse meta-targets instead of reliable meta-targets for classification datasets. However, this result was not mirrored for regression datasets, likely due to encountered complications. Furthermore, the current model's performance, using either reliable or diverse meta-targets, was evaluated by comparing the top-recommended DPP pipelines to those recommended by a popular DPP pipeline optimisation tool, TPOT. Though no statistically significant improvements were found, the results suggest that both the reliable and the diverse pipelines led to better performances than those of TPOT. When comparing the performance of the recommended pipelines from either the reliable or diverse subset of DPP pipelines, the results suggested that the reliable meta-target selection method can lead to a better subset of DPP pipelines than the diverse meta-target selection method since the reliable pipelines lead to slightly better results than the diverse ones.

1 Introduction

When training machine learning (ML) models, many vital steps must be optimised to achieve the maximum learning potential. Amongst these steps is dataset preprocessing (DPP). This task consists of applying several DPP methods to a dataset before training an ML algorithm so that the information within can be extracted most efficiently. Many DPP methods aim to enhance datasets by removing their shortcomings. These methods include data cleaning, scaling or reducing and resampling. However, finding an ideal string of DPP methods that are applied to a dataset is complex. The difficulty of this task is attributed to no universal pipeline existing that will reliably lead to the maximum performance for any dataset, since different datasets require different DPP methods. Furthermore, there is no obvious connection between dataset characteristics and the effect of DPP methods. This means, to determine the utility of a DPP method on a specific dataset, the effect on ML algorithm performance must be evaluated after fully training and testing an ML model. Finally, since many different DPP methods exist, the search space for the optimal configuration of different DPP pipelines is large. Combined, these factors lead to the complexity of DPP pipeline optimisation, which often requires much expertise and time to complete successfully. Besides relying on experience to intuitively design ideal DPP pipelines or employing grid search approaches that take considerable time, more advanced strategies exist. Such advanced strategies usually aim to solve this task efficiently and automatically. One such approach consists of training meta-learning models.

Meta-learning, or finding models and algorithms that can learn about learning itself, can be employed in an extensive range of applications. With the growing popularity of Automated Machine Learning (AutoML) in recent years, meta-learning is an ideal extension of this trend. However, unlike AutoML models, where an algorithm will use iterative ML training with directed approaches for optimising ML model hyperparameters, meta-learning relies on information about the performances of many previously evaluated ML models. With this, predictions about model performance for future tasks can be made.

Meta-learning systems are split into two major parts, an offline and an online setting. In the offline setting, information about previous ML model performances on various datasets is collected. This collected information is then gathered in a structure known as a knowledge base. By varying model hyperparameters for each learning task in the knowledge base, information on the effects of these hyperparameters is accumulated. This information can then be aggregated in meta-datasets. These meta-datasets take dataset information in the form of dataset meta-features as input and return predictions about hyperparameter performances as the target output and are used to train meta-models. Although the offline stage may take extended periods of time to complete, once the meta-model is trained, it can be used to make quick predictions about ideal model hyperparameters. The main requirement entails extracting novel datasets' meta-features and insert them into the trained meta-model. In the case of DPP pipeline recommendation, the targets relate to DPP pipeline performances for an incoming dataset.

Recently, researchers have used many methods for applying meta-learning to predict suitable DPP pipelines for novel datasets. These methods range from varying the target the meta-learner will predict (Bilalli, Abelló, Aluja-Banet, & Wrembel, 2018) to investigating whether using several smaller meta-models lead to better results than using a single larger meta-learner (Zagatti, 2021). The resulting meta-models from these studies generally report that strong DPP pipelines can be consistently recommended. However, it is common to encounter low meta-model accuracies when using this approach. Frye (2023), for instance, reported high DPP pipeline performance after obtaining DPP pipeline recommendations from a meta-learner; however, the actual meta-model accuracy was reportedly low. This suggests that the learning task of predicting likely DPP pipeline performance given task information is complex and not simple to solve. For this reason, the current project aims to investigate a novel method that could increase meta-model accuracies when making DPP pipeline recommendations.

The method that this project will investigate deals with the method in which meta-targets are selected. Common DPP pipeline meta-learners reduce the number of possible meta-targets to a subset of candidate pipelines. This step is required to avoid selecting ideal DPP pipelines from the set of all possible DPP method combinations. The standard approach to selecting DPP pipeline candidates follows a general DPP pipeline ranking technique based on average DPP pipeline performances of the available information obtained from previously run ML algorithms (Laadan, Vainshtein, Curiel, Katz, & Rokach, 2019; Zagatti, 2021). This approach is named the reliable method since the chosen pipelines achieve high average performances for any dataset. Choosing such meta-targets fills the subset with pipelines that will lead to decent dataset transformation, regardless of which is eventually recommended.

Though having reliable DPP pipelines in the subset of candidate pipelines is beneficial, the performances of these pipelines generally show very similar performances across different tasks, making it harder for learning algorithms to differentiate between them. This issue is expected to contribute to the low observed meta-learner accuracies for DPP pipeline recommendation. Additionally, since researchers have established that no universal pipelines exist that lead to maximal performance for any dataset (Jiao, Li, Chen, & Fei, 2020; Giovanelli, Bilalli, & Abelló Gamazo, 2021), it follows that no recommended DPP pipeline from a set of reliable pipelines can lead to a maximal performance for any dataset. Thus, an alternative approach for selecting DPP pipelines is proposed.

Whereas a reliable meta-target selection method may cast aside DPP pipelines that are specialists for only certain groups of tasks and not others, this project aims to find a set of pipelines that ensures these specialists are identified. This proposed meta-target selection method is named the diverse method and aims to obtain a set of more specialised DPP pipelines that each excel for different groups of tasks. Additionally, since DPP pipelines will be identified that will show more pronounced performance disparities between one another for a given task, diverse meta-targets are expected to lead to greater meta-learner accuracy when identifying ideal DPP pipelines.

1.1 Research Questions

The model created in this project will employ a diverse meta-target selection method to see if using it over a reliable approach can lead to better meta-model performances for both classification and regression datasets. This will be accomplished by using previously collected model evaluations, known as a knowledge base, to create meta-datasets that will be used to train a meta-learner. Instead of only predicting the best-performing DPP pipelines, the finished model will predict the performance of each DPP pipeline chosen by novel meta-target selection methods for a given task. Thus, the performance of the model can be determined based on its ability to predict model performance, as well as its ability to identify the best performance pipelines.

This project aims to answer two questions. The first refers to the meta-target selection method. A comparison of model performance will take place that will look at the meta-learners prediction performance when using diverse meta-target selection methods compared to reliable approaches used in the literature. Secondly, the model's ability to recommend well-performing DPP pipelines will be tested by comparing the recommended pipelines to those recommended by a commonly used DPP pipeline recommendation tool. The two research questions are summarised as follows:

Q1. Can diverse meta-target selection outperform reliable meta-target selection by improving classification and regression meta-model performance?

Q2. Will the DPP pipelines recommended by the current model lead to a better model performance than those recommended by current state-of-the-art DPP pipeline recommendation systems?

1.2 Thesis Outline

This thesis will first introduce the background behind the need for DPP meta-learning methods and AutoML systems to solve this issue. Subsequently, previous DPP recommendation systems will be introduced briefly. Following the introduction to the source material, the development process involved in creating the current model will be presented. This overview includes information on the construction of the knowledge base, initially created by Frye (2023), the meta-dataset creation process, including dataset meta-feature extraction and the different meta-target selection procedures. Finally, the training and testing of this model will be analysed, focusing on answering the research questions mentioned above.

2 Background

Machine Learning (ML) is a standard procedure in the current academic and industrial climate. Many companies see the potential in applying ML algorithms to aid in many situations, including risk analysis, industrial fault detection, translation and social media recommendations. In general, an ML model is used to learn the underlying patterns of a dataset and make predictions about future inputs within the dataset's domain. Contributing to the popularity of ML techniques, the field of ML is constantly growing. Where early ML approaches could only hope to solve linearly separable classification tasks (Haykin & Network, 2004), modern approaches can learn extremely complicated tasks ranging from classification of tabular data to detailed image classification and online speech comprehension (Lu, Liaw, Wu, & Hung, 2019; Kano, Sakti, & Nakamura, 2021).

Though the competence of ML models increases with each passing year, the process of successfully training and deploying ML algorithms to obtain strong model performances is not straightforward. It requires a structured approach rather than just applying a powerful ML algorithm. Furthermore, specific steps need to be fulfilled to enhance the success rate of such projects. These steps involve understanding the data, cleaning the data, finding the ideal model and eventually evaluating the model's performance. Only by applying each step can one train models successfully.

Dataset preprocessing is one of the most vital steps of any machine learning project. It refers to the manipulation and transformation of datasets through various means, each helping to account for a shortcoming of the dataset. However, adequate data preprocessing is highly dependent on developer expertise as well as the amount of time invested in the process. Even with much experience, finding optimal data preprocessing solutions can be time-consuming. To handle this vital stage in the ML process more reliably and efficiently, a need for automation of data preprocessing emerged.

With a demand for effective solutions for data preprocessing, a push toward automation was achieved by utilising Automatic Machine Learning (AutoML). AutoML is a modern approach to ML that aims to, as the name implies, automate parts of the ML protocol and can be used to determine ideal settings for individual sections or entire ML processes. This is achieved by running and evaluating several ML models and using optimisation techniques to adapt model hyperparameters across several iterations. Another approach to obtaining ideal model settings is to make use of meta-learning. Here, information based on a collection of previously evaluated ML algorithms is gathered and used to predict how ML algorithms will perform for unseen datasets.

The following literature review will touch on the points mentioned above in greater detail. Initially, the important steps within a machine learning project will be introduced (Chapter 2.1), followed by the importance and types of data preprocessing methods (Chapter 2.2). Finally, AutoML systems are discussed as a way to tackle DPP pipeline recommendation in Chapter 2.3.

2.1 Machine Learning Process

Different approaches may produce different results when developing an ML model. For example, a novice may employ the most powerful deep neural network, hoping its sheer learning capability will be enough to solve the given task successfully. Though learning may occur, depending on the general difficulty attributed to the dataset, a simpler model could have solved the task in a faster, more explainable manner. On the other hand, a slightly more experienced developer may wish to select an ideal learning algorithm based on the dataset itself, and another may have analysed the dataset's characteristics and determined that feature engineering is required. Though these latter approaches are more desirable than instinctively resorting to powerful models, much performance disparity is reported that can be traced back to differing practices (Lee & Giraud-Carrier, 2014; Chandrasekar &

Qian, 2016; Alam & Yao, 2019; Fischer et al., 2020). In an effort to make sure that ML projects adhere to best practices and obtain high-performing models, process models have been designed that aim to standardise specific procedures. Thus, process models such as CRISP-DM (Shearer, 2000), and more recently CRISP-ML(Q) (Studer et al., 2021) have been used to outline essential steps within any ML project that should be adhered to, ensuring optimal model performance. In Chapters 2.1.1-2.1.3, the essential steps relevant to this project will be introduced.

2.1.1 Data Understanding

The first step to any ML project is understanding the dataset or task one aims to solve. Shearer (2000) defines four steps that can be completed in this stage: collecting, describing and exploring the data and verifying its quality. When the data is of decent quality, the information obtained from the data understanding section will be used to structure the subsequent step.

2.1.2 Data Preprocessing

After the data has been explored, its flaws are targeted. Dataset Preprocessing involves cleaning, transforming and otherwise manipulating the data to ensure that the model used in the subsequent stage can achieve the best performance possible. This manipulation is accomplished through various methods applied to the dataset, each fulfilling a different function. Which methods should be applied is determined from the previous processing step. Since data preprocessing (DPP) is such a vital part of the current project, the intricacies and importance of this step will be explained in greater detail in Chapter 2.2.

2.1.3 Modelling

The heart of any ML project is the training of ML models. Many different techniques can be employed depending on what a project is aiming to do. For example, models that can help find underlying patterns, such as clustering methods, may be employed for data mining projects. In contrast, trainable supervised learning algorithms are used predominantly in ML scenarios. In any case, ideal models are selected based either on the project's goal or which model best suits the data.
Once a model is selected, tuning its settings to the dataset ensures the best results. Hyperparameter optimisation, for instance, refers to the search for the correct configuration of the chosen model. This crucial step can decide between a successful model and one unable to achieve desired results (L. Yang & Shami, 2020). There are many different ways hyperparameters can be selected. Manual search, for example, can be an effective and time-efficient method of selecting well-performing parameters; however, years of experience correlate strongly with the success of the resulting models (Anand, Wang, Loog, & van Gemert, 2020). Naturally, developers with less experience often rely on methods with a degree of automation to make up for such disparities. Besides using grid searches, which may take unrealistic amounts of time depending on the hyperparameter combination search space size, other automatic searches employ more intelligent strategies to find suitable model settings. Such methods include Bayesian optimisation (Wu et al., 2019), reinforcement learning (S. Chen, Wu, & Liu, 2021), and genetic algorithms (Alibrahim & Ludwig, 2021), each shown to yield reliable results. In contrast, randomly selecting model hyperparameters is unreliable for achieving high-performing models (Bergstra & Bengio, 2012), highlighting the need to employ experienced developers, invest large amounts of time, or use specialised automated methods to achieve the goals which were set.
The models employed in ML are usually trained to solve tasks via supervised learning. To train a supervised learning model M, a dataset D consisting of a set of input and output pairs $d_i = \{x_i, y_i\} \in$

$D, i = 1, ..., I$ is required. Over the learning procedure, each of the inputs will be presented to the model, each producing a predicted value $y_i^{pred} = M(x_i)$ and, following a review of the accuracy of the prediction y_i^{pred} compared to the target value y_i, the model's internal parameters will be adapted. Two major categories of ML models are generally employed to solve supervised learning tasks. The choice depends on the task a developer aims to solve and the kind of target value used in the dataset.

Classification Models Classification tasks predict the class membership of a data point, whereby the membership is given by the target value y_i in the training set. This target value corresponds to a particular class included in a set of classes $y_i = c_j \in C, j = 1, ..., J$. The target values are categorical for classification tasks, referring simply to the class assigned to an input within. Some practical applications of classification models are Email spam detection (Chirita, Diederich, & Nejdl, 2005), production-line fault detection (Basnet, Chun, & Bang, 2020) and sentiment classification (Hadwan, Al-Sarem, Saeed, & Al-Hagery, 2022).

Regression Models Unlike the classes predicted in a classification task, the target values in a regression task are continuous. Rather than predicting the target membership y_i, the prediction is attributed to a numerical value associated with the target. The most basic regression model is the linear regression model. This model is trained by fitting an N-dimensional hyperplane within a space according to an N dimensional input x_i. Since regression aims to predict the size of a specific value, it can be used in a wide variety of practical applications, for instance, for temperature prediction (Ghosh, Satvaya, Kundu, & Sarkar, 2022), financial forecasting (Chou & Nguyen, 2018) and time series modelling (Babii, Ghysels, & Striaukas, 2022).

Model Assessment After selecting and training an ideal model, an assessment is required to determine whether the training was successful. This evaluation is achieved by determining a model's ability to accurately predict a subset of the data within a dataset not used during training. This is done by utilising a train-test split before the model's training. Here, a section of the data used to train the model is strictly reserved for evaluation, allowing developers to understand how the trained model will perform if used actively on novel incoming data.
The quality of predictions on the testing set indicates how the model would perform if employed in a real-world setting. This performance can be evaluated via several measures depending on the type of task that the model is learning. For example, in the case of classification tasks, where a predicted target can either be correct or false, one standard method is to evaluate the model's accuracy.

$$Accuracy = \frac{NumberOfCorrectlyClassifiedTargets}{TotalNumberOfTargets} \tag{1}$$

The Accuracy measure can give a decent idea of how well a model performs. However, in some cases, this measure can be misleading. For instance, if a significant class imbalance exists and a model mistakenly learns to predict the majority class label consistently, the Accuracy measure could still indicate good performance. Therefore, an alternative measure is usually preferred to avoid falsely inflated performance evaluations. The F1-Score, for example, is a performance measure that penalises a model for making wrong predictions.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2}$$

With a dataset containing dichotomous targets (for simplicity, inputs are given either positive or negative labels), *Precision* refers to the ratio of true positives over the total number of inputs given the

positive classification. A model with high Precision might only find some positive targets but is unlikely to incorrectly classify an item as positive. *Recall* is the second part of the F1-Score and refers to the ratio of true positively classified inputs over the total number of positive inputs. A model with high Recall can correctly identify the positive cases in the data, whereas a model with low Recall may miss a portion.
If the ML algorithm solves a regression task, a classification performance measure will not be suitable. Instead, distance measures are used since the targets are continuous values. One way of calculating the distance between the model's prediction and the actual value is to employ the mean squared error (MSE)

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n} \tag{3}$$

where y_i is the target value for input i and $\hat{y}_i$ is the corresponding predicted value. The MSE can be extended by adding the root to the measurement, making the RMSE.

$$RMSE = \sqrt{MSE} \tag{4}$$

Like the MSE, the RMSE value is used to determine the level of goodness that the fit of a trained model was to the actual targets being predicted. The main benefit of using the RMSE measurement over the MSE measurement is the ease of interpreting the resulting value. An MSE value penalises large deviations from the target value, resulting in a large MSE value for greater distances and lower interpretability. RMSE, on the other hand, removes this inflated distance score by reverting the scale of the MSE score down to the scale of the target, increasing interpretability.

2.2 Dataset Preprocessing Methods

It is expected that, when working with real-world data, not all records in the data are helpful or even ready to be used. Some features may contain missing values due to data collection issues, and others may be less critical and do not contribute significantly to predicting the targets. Dataset preprocessing (DPP) methods are transformations a user can perform on a given dataset to account for shortcomings that the dataset may possess. To illustrate the importance of using DPP methods optimisation in combination with tuning learning algorithm hyperparameters, Giovanelli et al. (2021) aims to optimise an ML model for 100 iterations. Here, over the first 50 iterations, the optimisation process that was used only focused on selecting ideal hyperparameter configurations. As seen in Figure 1, the model's performance plateaus after around five iterations. This result emphasises that after a good combination of hyperparameters was found, no further significant improvements could be made. Then, after the 50th iteration, the optimisation process included DPP method optimisation. We can see from the plot that accuracy increases once again after DPP methods are employed. The results highlight the importance of the interplay between hyperparameter optimisation and DPP pipeline selection and help explain why researchers spend a considerable amount of time within a project on finding the correct way to process the data (Munson, 2012).
DPP methods can help extract the largest potential from a dataset and can be sorted into several categories according to what issue a method aims to solve (García, Ramírez-Gallego, Luengo, Benítez, & Herrera, 2016; Li, 2019). In this paper, the following four categories will be used to identify the underlying functionality of a DPP method: Data Cleaning, Data Transformation, Data Reduction and Data Resampling. Each category will be introduced in Chapters 2.2.1-2.2.4. Of course, datasets usually require the employment of multiple DPP methods. This is usually achieved by combining DPP methods into DPP pipelines, a process introduced in Chapter 3.1.2. Since this project deals with

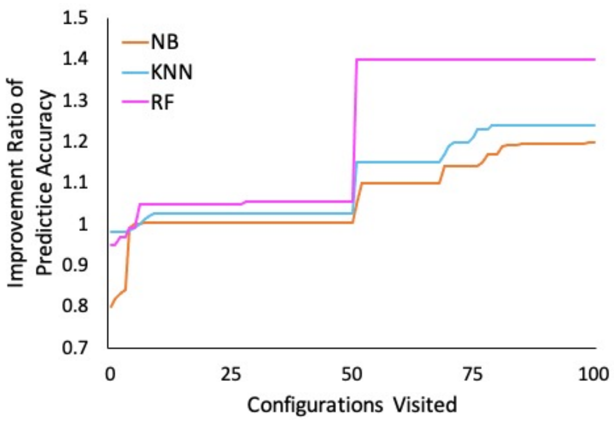


Figure 1: Accuracy increase of ML algorithms (Naive Bayes:NB, K-Nearest Neighbours:KNN and Random Forest:RF) during the optimisation process. Only hyperparameter settings are optimised between iteration 0 and iteration 50, leading to a plateau. After iteration, 50 DPP methods are also included in the optimisation process, leading to further increases in model accuracy (Graph taken from Giovanelli et al. (2021)).

tabular data and not image, time-series or language data, only preprocessing methods relevant to the project will be introduced. For this reason, many preprocessing methods that are useful for different ML domains, such as several augmentation methods and other image, time series and text-based processing methods, will be left out.

2.2.1 Data Cleaning

Data cleaning methods aim to fix imperfections in the dataset that may include missing values or outliers. Missing values are usually a result of faulty data collection and can be a major issue for datasets. One way to deal with missing values is to remove any data point that contains a missing value. This technique allows the remaining data points to consist of complete recorded values; however, if many values are missing or the data is not abundant and hard to obtain, decreasing the size of the dataset may be detrimental.

An alternative solution is to impute missing values through various techniques. These techniques include simply replacing missing values with a corresponding feature's mean, median or most frequent values. Additionally, more complex statistical methods exist, such as linear regression or least squares models, that are used to predict the missing values (Lin & Tsai, 2019). Since many ML algorithms require datasets to include zero missing values in order to run, missing value imputation can be considered a mandatory step in dataset preprocessing.

Besides faulty data collection methods leading to missing values in the data, other artefacts can appear as outliers. Outliers are data points with values that lie far away from the values of other data points. In some cases, a large distance may reflect the ground truth, such as recording the age of an individual who has reached the age of 120; however, in most cases, these outliers result from inaccurate measurements (Li, 2019). To deal with such instances, one must first identify outliers so they can be removed. For smaller datasets, this can be done using simple distance measures. For example, using box plots and interquartile range or relying on the standard deviation of specific attributes are popular and reliable methods in these cases (Ben-Gal, 2005). On the other hand, cluster-based outlier detection might be the preferred technique for larger datasets and produce similarly reliable outcomes while being more efficient (Ben-Gal, 2005; Duan, Xu, Liu, & Lee, 2009).

Another simple form of data cleaning is to remove any features of instances that can be assured to not aid in the training. Examples of such parts of data are duplicate rows or columns. Suppose a row or column of a dataset is a duplicate of another. In that case, no additional information can be obtained by including it in the final dataset and may even be dangerously misleading in the modelling process (Brownlee, 2020). In such a case, it is best to remove these rows of columns in the interest of simplicity. Data cleaning methods are usually the first step into dataset preprocessing and make datasets easier to work with, less error-prone and more interpretable.

2.2.2 Data Transformation

As the name implies, data transformation methods transform data and make it more accessible for learning algorithms to interpret. Furthermore, since many ML algorithms prefer working with numbers over other variable types, such as categorical features represented as strings, using methods to transform such attributes into more suitable types can be considered another mandatory step during the data processing phase.

There exists a large variety of feature encoders that can each affect a dataset differently. One of the most popular encoding methods is the One-Hot-Encoder. This encoder will create a sparse set of features such that one feature for each category within an attribute is constructed. This method is viable for datasets with higher cardinality since the transformation can lead to more easily optimised datasets. However, should a large number of categorical features exist within a dataset, the sparse nature of the returned transformation that the One-Hot-Encoder produces can cause the dimensionality of the datasets to explode (Seger, 2018). In such instances, encoders that, by design, aim to maintain a lower dimensionality after encoding, such as the hashing encoder or target encoder, may be the preferred option ("A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems", 2001; Seger, 2018; Cerda & Varoquaux, 2019; Gupta & Asha, 2020).

Just as categorical features can be encoded into numerical ones, the same can be applied to continuous features. *Discretisation* is a DPP step aiming to simplify datasets. They help boost explainability by transforming continuous variables into countable numerical ones. This widely popular approach not only helps to represent features more precisely but can also help speed up and increase the accuracy of ML models (H. Liu, Hussain, Tan, & Dash, 2002; Elhilbawi, Eldawlatly, & Mahdi, 2021). Discretisation is achieved by splitting continuous variables into meaningful *bins*, which can be done simply by manually or automatically defining bin sizes or using other ML models, such as using decision trees to split up the data (Li, 2019). Of course, more advanced approaches such as MDLP and Chi2 can be employed to discretise features, and these have been linked with consistently higher accuracy at the cost of complexity (Lavangnananda & Chattanachot, 2017).

The final notable step that belongs to data transformation is data normalisation. This essential DPP technique involves centring data around specific anchor points or within common ranges and primarily aims to increase the potential impact some attributes have on the dataset (Li, 2019). Data normalisation steps are highly effective in boosting model performance by helping to minimise bias from features that may overshadow others in magnitude. Essentially, this attribute centring allows each feature to contribute equally to the task at hand (Singh & Singh, 2020). However, though this technique levels the playing field, it does not guarantee that all features will end up as necessary as the rest. Mean and standard deviation-based normalisation methods such as the mean-centred (MC), z-score normalisation and power transformer techniques, and other normalisation techniques like min-max scaling and Hyperbolic Tangent scaling belong to the notable normalisation methods and can be found in many DPP solutions (Sharma, 2022).

2.2.3 Data Reduction

The datasets nowadays are increasingly large; however, an extensive number of attributes does not necessarily correlate with high model accuracy. Thus, removing features that do not contribute strongly to the prediction performance is often beneficial in the pursuit of less complex and more efficient models (Khalid, Khalil, & Nasreen, 2014). Removing redundant features is known as feature selection. It aims to employ methods that help select only those features that contain the most information about the model while keeping it as simple as possible.

A further reason for striving to keep the size of a dataset small is to avoid the curse of dimensionality. This *curse* can be traced back to the mathematical account that, as the dimensionality of an object increases, so too does the space within the object. Figure 2 illustrates this. Here we have the probability density functions regarding the distance between the corner of an *N* dimensional hypercube and a random point within. It can be seen that the distance of a randomly inserted point within the hypercube to the corner of that hypercube becomes constant as the dimensionality increases. Statistically, the probability of a randomly allocated point being close to the corner reaches 0 as the dimensionality of the hypercube reaches infinity. The reason for this convergence can be understood in that the hypercube's volume is concentrated around the centre. The volume of the regions close to the corners, on the other hand, reaches 0 with increasing dimensionality (Crespo Márquez, 2022). Concerning ML models, the curse of dimensionality implies that a model will benefit from a lower-dimensional dataset since there is less space for the data to occupy. Thus, better relational information can be learnt since the data points will generally be closer together (Steinbach, Ertöz, & Kumar, 2004). Besides the distance increase between points of a high-dimensional dataset, the curse of dimensionality is also linked to the number of data points required for proper training. The number of data points needed to represent the model correctly will increase exponentially as the number of dimensions increases (Bühlmann & Van De Geer, 2011).

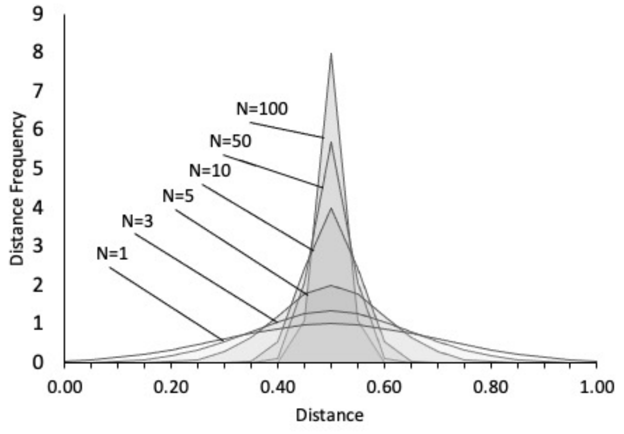


Figure 2: Probability density functions of the distance between the corner of a hypercube and a random point with an increasing number of dimensions (Graph taken from (Crespo Márquez, 2022).

Numerous methods can be used to reduce the dimensionality of datasets. One of the most popular methods is to conduct a principal component analysis (PCA) (Li, 2019). During PCA, new attributes (principal components) are constructed along the dataset's most prominent variance axes. From here, the top *N* principal components represent the top *N* directions in which the data is spread (Abdi & Williams, 2010). It is then up to the researcher to determine how many principal components should remain after PCA, allowing many attributes in the original dataset to be reduced to a much smaller number. Besides PCA, various methods exist to reduce the dimensionality of datasets, including K-Means Clustering (Napoleon & Pavalakodi, 2011), Linear Discriminant Analysis (LDA) (Xanthopoulos et al., 2013), and other clustering methods and can be an essential tool when working with ML models.

2.2.4 Data Resampling

Real-world data is often unbalanced, meaning that the number of targets in one class may be on a different scale than targets of other classes, which can be simply due to the nature of the task. For instance, when working with medical data, there is often much more available data that describes healthy patients than patients with rare diseases. As a result, problems arise when using learning algorithms to identify targets from the minority class. Often, a high rate of false negative predictions can result when a dataset has a severe class imbalance (Longadge & Dongre, 2013). To reduce the effect of class imbalance, processing methods can be employed to even out the asymmetry of classes by means of data resampling methods.
The simplest way a dataset can be balanced is to undersample the data. Here, a portion of instances from the majority class is removed to lower the level of class imbalance in the data. Undersampling, however, results in a loss of information and can only be recommended in some instances (Batista, Prati, & Monard, 2004; García et al., 2016; Dal Pozzolo, Caelen, & Bontempi, 2015). Other techniques aim to rebalance datasets by oversampling, a form of data augmentation that involves creating new data points belonging to the minority class. The Synthetic Minority Oversampling Technique (SMOTE) is an example of such a method (Fernández, García, Herrera, & Chawla, 2018). It aims to generate new instances by constructing them between existing pairs of instances (see Figure 3). This technique has been shown to produce balanced datasets that do not lead to unwanted outcomes such as overfitting, which may result from oversampling by just duplicating examples from the minority class. Overall, resampling is an important step that can improve dataset performance and is a vital measure in fields where the likelihood of attaining balanced datasets is low.

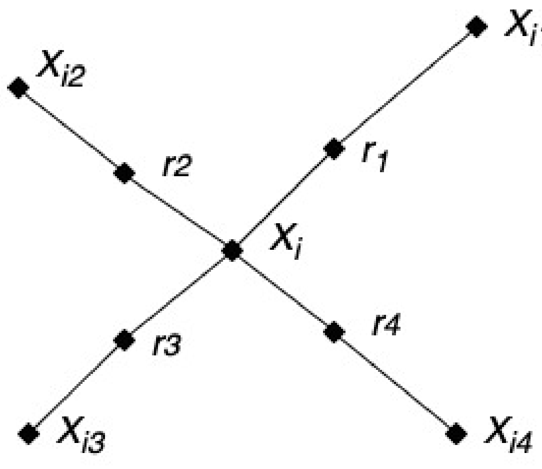


Figure 3: New data points, denoted *r*, are generated along the axes connecting existing data points, denoted *x*) (Image taken from Fernández et al. (2018)).

2.2.5 DPP Pipelines

There are a large number of ways that a dataset can be preprocessed, with each method accounting for different issues. Usually, more than one DPP method is required to ensure the dataset is optimally prepared. Thus, a DPP pipeline can be created by combining multiple DPP methods into a chain of DPP methods that transform the dataset sequentially.

Naturally, the order in which DPP methods occur within a pipeline affects how well a dataset will be transformed. Testing the resulting accuracies of different DPP method type pairs, Giovanelli et al. (2021) found that specific orderings were more promising than others. Specifically, they found that data cleaning and encoding methods are most effective at the beginning of pipelines with data transformation steps in the middle, followed by feature creation and resampling methods (see Figure 4). Since we can string different DPP methods together in an ideal order, the question of an ideal, universal pipeline arises. Would it not be possible to combine all the DPP methods to improve model accuracy into one large pipeline and avoid time spent on dataset-specific optimisation?

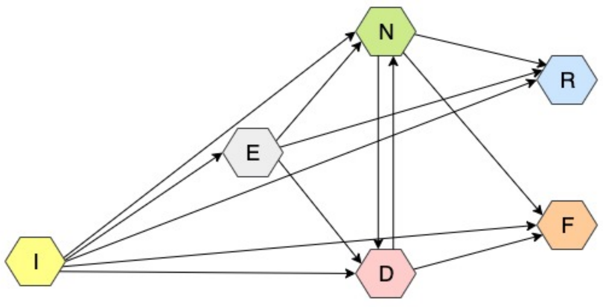


Figure 4: Optimal consecutive DPP steps (I - imputation, E - encoding, N - normalisation, D - discretisation, R - rebalancing/resampling, F - feature engineering). Image taken from Giovanelli et al. (2021).

Though each dataset will likely benefit from preprocessing, not every pipeline will benefit each dataset (Giovanelli et al., 2021). Various studies have shown that no universal pipeline will achieve the highest performance for every dataset (Quemy, 2019; Jiao et al., 2020; Giovanelli et al., 2021). Besides different datasets requiring specific treatments and redundant methods unnecessarily increasing complexity distortion, the choice of ML algorithm to be applied after processing has a considerable influence on which DPP steps will lead to better results (Bilalli et al., 2018). For this reason, it remains a challenging task to design a fitting pipeline for a given dataset. However, as hyperparameter optimisation can be automated, so can DPP pipeline selection. The subsequent section will discuss how automated ML can help search for ideal DPP pipelines.

2.3 Automated Machine Learning

Automated Machine Learning (AutoML) was first defined as ”software capable of being trained and tested without human intervention”. It was initially the concept of the 2015 ChaLearn challenge, where participants were instructed to design such systems (Guyon et al., 2015). Though there were instances of automated systems created before the ChaLearn challenge (Hutter, Hoos, & Leyton-Brown, 2011; Thornton, Hutter, Hoos, & Leyton-Brown, 2012; Komer, Bergstra, & Eliasmith, 2014), the popularity surrounding AutoML has increased since then to the point where using these systems is more commonplace than ever before. Reasons for the popularity increase are likely due to the increasing availability of computing power and the problems mentioned earlier: creating a well-performing model is a tedious process that requires either large amounts of time or high levels of experience (He, Zhao, & Chu, 2019). Having a ready-to-use solution that can reliably produce well-trained models, even for novice developers, has been a noteworthy milestone in the world of ML and has spurred the development of AutoML systems in recent years (Nagarajah & Poravi, 2019).

2.3.1 Types of AutoML Schemes

Given that the previously mentioned definition of AutoML systems is broad, a substantial collection of different types of automated models exist. For instance, complete AutoML systems are available that span the entire ML process, including model selection, hyperparameter optimisation and DPP pipeline selection (Laadan et al., 2019; Olson et al., 2016). In addition, an AutoML system can also refer to automating individual steps within the ML process (Escalante, 2020). Examples of such systems are, those that automate model selection (Rice, 1976; C. Yang, Akimoto, Kim, & Udell, 2018), hyperparameter optimisation (Wu et al., 2019; S. Chen et al., 2021; Alibrahim & Ludwig, 2021; Chatterjee, Bopardikar, Guerard, Thakore, & Jiang, 2022), Combined Algorithm Selection and Hyperparameter optimisation (CASH) (Chatterjee et al., 2022; Thornton et al., 2012), or neural network architecture search (Wistuba, Rawat, & Pedapati, 2019).
Since the different tasks that an AutoML system aims to solve are closely related, making it challenging to distinguish between them, Z. Liu et al., 2019 proposed a unifying view that characterises a three-level formulation of AutoML schemes depending on the level of automation (see Figure 5). The α-level scheme refers to a search for a functional mapping of an input to an output. To reach this goal, no level of automation is required, meaning that this function could be heuristically designed by either building rule-based classification systems, setting the weights and hyperparameters of learning models manually or via a random procedure. An α-level solution is a model that makes a prediction after having been trained on a particular dataset and is the result of the more commonly found models that are acquired using higher level schemes.

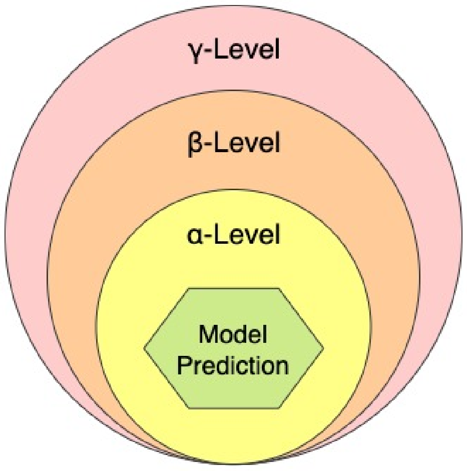


Figure 5: The three AutoML schemes as proposed by Z. Liu et al. (2019). The lowest level, the α-level, consists of an estimator or functional mapping that can produce a prediction, given an input. The goal of each model on a subsequent level is to produce an ideal model on the prior level given based on many performance evaluations of lower-level models. For instance, a β-level model must evaluate the performance of many α-level models to return a finished model capable of producing an optimal prediction performance. On the other hand, a γ-level model evaluates the performance of many β-level models to recommend an ideal β-level model, configurations that the ideal β-level model would possess that will ultimately lead to an optimal α-level model.

The β-level scheme requires a level of automation to search for an optimal α-level model. To achieve this, an automatic procedure is used to test multiple α-level models to determine which model weights

or hyperparameters lead to good performances. This solution, for example, can be a trained ML algorithm that has learnt to find a decision function using stochastic gradient descent. However, aβ-level scheme can also entail the search for ideal ML algorithm hyperparameters and DPP pipelines by iteratively evaluating the performance several α-level models using different model configurations on the dataset in question.
The final level, the γ-level scheme, refers to meta-learning, whereby a knowledge base is exploited to make predictions about the performance of ML models with regard to a specific measurement, including but not limited to ML algorithm performance, hyperparameter configuration performance, DPP pipeline effects or even ML model speed. A *knowledge base* is a collection of results of previously trained and evaluated β-level algorithms that can be used for informed decisions and make recommendations on which β-level algorithm to use for a particular task. To summarise, each level of the AutoML schemes return the model used during the realisation of the level below (Escalante, 2020). Thus, the difference between current AutoML models and whether they fulfil a β or γ-level scheme can be determined by considering the origin of the model's knowledge. The results are either derived from the dataset itself or information regarding previous models' performance.

2.3.2 Meta-Learners

Meta-learners solve a γ-level AutoML problem and thus aim to recommend ML models with ideal model hyperparameters (Feurer, Springenberg, & Hutter, n.d.). As previously mentioned, meta-learners require a knowledge base that can be used as a resource to help produce model recommendations. This knowledge base can be collected individually or extracted from existing meta-data repositories (Vanschoren, van Rijn, Bischl, & Torgo, 2014). Within a knowledge base, performance evaluations are collected after training ML algorithms on a set of prior ML tasks $t_j \in T$. For each task, the performance of models is recorded after varying model configurations $\theta_i \in \Theta$. In a meta-learning system, the performance differences are attributed to the variation of model configurations being performed. It is these performance differences attributed to various configurations that are to be learnt by the meta-learning system. For example, if the meta-learner's goal is to recommend ideal ML algorithm hyperparameters, the configurations within its knowledge base would refer to different hyperparameter compositions. For each different configuration applied to each of the tasks, a performance measure $P_{i,j} = P(\theta_i, t_j) \in \mathbf{P}$ is collected. The goal of a trained meta-learner is that it will be able to understand how a model will perform given a new task t_{new} and a particular set of model configurations (Vanschoren, 2019). If its task is to recommend ideal hyperparameter combinations, the combinations with the highest predicted model performance will be recommended.
A meta-learner will make predictions about ideal model configurations given t_{new} by exploiting the information about previous tasks contained within the knowledge base. Similarities between the new task and the previous tasks can be identified by extracting meta-features that characterise each task. Many meta-features can be extracted from a task ranging from straightforward to complex. Starting with simple meta-features such as those describing elements of the dataset, including the number of attributes, classes, outliers and missing values, to more involved meta-features that analyse the feature independence, task complexity, feature importance, sparsity, and quality (Krouwer & Schlain, 1993; Peng, Flach, Soares, & Brazdil, 2002; Pfahringer, Bensusan, & Giraud-Carrier, 2000).
Meta-features are descriptors of tasks and function in vector form. Each task $t_j \in T$ can be described by the meta-feature vector $\vec{m}_{t_j} = (m_{j,1}, ..., m_{j,K})$ of K meta-features. These meta-feature vectors are later used as inputs in meta-datasets, with the targets being the performance evaluations for a model configuration on that particular task $P(\theta_i, \vec{m}_{t_j})$. Meta-datasets are then employed as training sets to train the resulting meta-learning system. The learning task relies on the expectation that if two tasks

seem similar based on the extracted meta-features, then the configurations should lead to similar model evaluations. From here, the relationship between the task's meta-features and the effects of the specific configurations can be learnt using different ML algorithms known as meta-models. In the literature, this method has proven an effective way to make quick predictions for ideal model configurations on new tasks (Brazdil, Giraud-Carrier, Soares, & Vilalta, 2009; Peng et al., 2002; Garouani et al., n.d.; Rivolli, Garcia, Soares, Vanschoren, & de Carvalho, 2022).

2.3.3 Advantages and Disadvantages of Meta-Learning

There are significant benefits to using a γ-level over a β-level AutoML system. For one, a meta-learner consists of two phases: online and offline. The most time-consuming section is the offline phase, where the information about previous tasks is collected, and meta-models are trained. On the other hand, the online phase is where the meta-learning system is applied to produce predictions about a new task. Here, all that is required from the system is the extraction and insertion of meta-features of the new task into the previously trained meta-model to obtain configuration predictions. Furthermore, since any model training has been completed offline, the speed at which the predictions can be made in the online phase of the developed models is very fast (Dyrmishi, Elshawi, & Sakr, 2019). In contrast, β-level methods complete the entire task in the online phase and are given a time budget that can be expended during the optimisation processes. Therefore, the more time one invests in finding optimal model solutions, the greater the chances of obtaining a positive outcome. It can thus be rather time-consuming to find ideal fitting solutions for a given task. A second benefit of using a γ-level model is that the information within the knowledge base used to train the resulting models can be continuously extended over time, improving its prediction accuracy (Dorfman & Tamar, 2020). On the other hand, the performance of β-level models depends predominantly on the systems put in place on the model's release, only increasing performance when newer updates to the software are available.

Besides the benefits, some substantial drawbacks exist when using γ-level over β-level models. For instance, a knowledge base requires information on the performance of previously run ML algorithms, ideally using many heterogeneous datasets. However, obtaining large numbers of unique datasets can be challenging, especially if a specific dataset domain is required (Griffiths et al., 2019). Not only is the training task harder with smaller training sets, but this lack of data often leads to generalisation difficulty and overfitting (Rajendran, Irpan, & Jang, 2020; Yao et al., 2020). To counter this problem, dataset augmentation can be applied to increase the number of datasets (Frye, 2023). This drawback does not exist for β-level models since the calculations are independently performed on each new task, and thus no dataset augmentation is required.

2.4 Related Work

When predicting DPP pipelines using AutoML, a sizeable amount of literature has been collected over the years. One of the influential and most often referenced models is the Tree-based Pipeline Optimisation Tool (TPOT) designed by Olson et al. (2016). Like many earlier AutoML models, this system aimed to simultaneously automate most of the ML pipeline, excluding data cleaning (see Figure 6). The model generates parameters via a genetic algorithm over several generations, including feature selection methods, feature transformation methods and models with different hyperparameters. The fitness of each tree structure in a generation is determined by recording the accuracy measure of the desired task by sending it through the tree structure and evaluating the final model contained in the structure. Although the process takes considerable time to run a model for each candidate tree

structure, this β-level AutoML method gained much recognition over the years due to its reliability in achieving high accuracies with little to no need for expertise in the topic.
Following the success of TPOT, several other approaches were designed to improve some of the shortcomings that TPOT possesses. For instance, to search for full pipelines, B. Chen, Wu, Mo, Chattopadhyay, and Lipson (2018) designed a similar model that is also trained with a genetic algorithm. As a result, their Autostacker model outperforms TPOT in accuracy and speed by using more advanced ensemble methods and including the search for optimal data cleaning methods in the optimisation process. Since then, further progress has been made by increasing automation and using γ-level meta-learning strategies to determine good ML pipelines. Though creating models that predict complete ML processes, including DPP pipeline selection and hyperparameter optimisation, is still actively pursued (C. Yang, Fan, Wu, & Udell, 2020), the ML community has increasingly focused on automating individual steps of the ML pipeline.
To this extent, the meta-learning model PRESISTANT was designed to recommend a set of DPP pipelines, given new datasets. Furthermore, PRESISTANT aimed to produce recommendations that achieve similar performances to pipelines created by experienced data scientists (Bilalli et al., 2018). This γ-level meta-learner uses the impact that different DPP methods have on the performance of a selection of ML algorithms, measured as the difference in accuracy before and after the transformation, to determine the effectiveness of different DPP methods. Each evaluation was then categorised into either causing a positive, neutral or negative impact. A pool of candidate pipelines was then selected using a rule-based approach. Finally, the predicted overall impact for the remaining pipelines was predicted, and the system would recommend the pipeline with the highest positive impact. The results showed that a meta-learning technique is a viable solution for DPP pipeline recommendation since the recommended pipelines, as expected, performed similarly to those designed by a group of experienced data scientists.

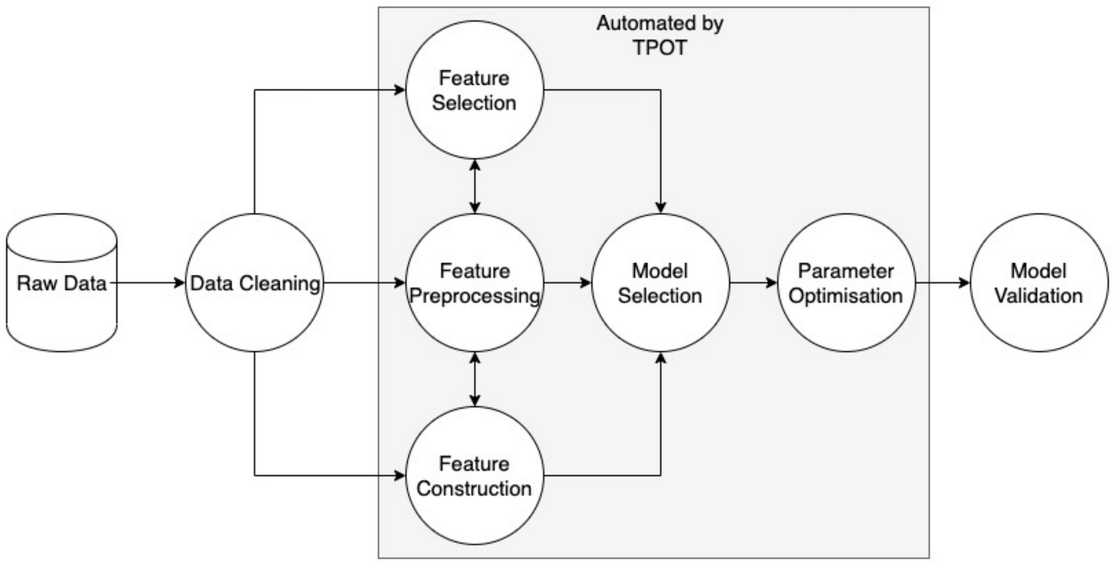


Figure 6: Overview of automated steps tackled by TPOT according to Olson et al. (2016).

Several similar models were also created, each possessing slight variations during the training or evaluation collection process. RankML collected a knowledge base of DPP pipelines and ranked them according to the accuracy scores obtained from model predictions (Laadan et al., 2019). From here, the top *N* were used to train individual ML models to determine which DPP pipeline returns the highest accuracy. Again, the average accuracy was greater than the scores achieved by previous DPP pipeline recommendation models, including TPOT. To reduce online prediction time by avoiding multiple training models in an online setting, Zagatti (2021) designed a meta-learner trained to predict the single best DPP pipeline for a given dataset. A dataset was crafted using meta-features by ranking each DPP pipeline in the knowledge base by accuracy. Each instance was assigned the best-performing DPP pipeline obtained from selecting the top-performing pipelines in the knowledge base constituting a set of DPP pipeline candidates. This step was improved by changing how the DPP pipeline candidates were selected. Instead of ranking the DPP pipeline evaluations in the knowledge base ordinally, Frye (2023) performed a Scaled Ranking procedure, where the distance between rankings was proportional to the accuracy measures and scaled between 0 and 1. As a result, each of these models could provide good DPP pipeline recommendations that achieved high accuracies for new datasets quickly.

3 Methods

The current project aims to obtain a trained multi-output regression meta-model to predict the goodness of DPP pipelines for new datasets. Figure 7 shows the methodological approach to achieve this goal. It consists of data collection, followed by the creation of meta-datasets and finally, the training of a meta-model using these datasets.

An AutoML model of type γ performing meta-learning based on previous knowledge will be employed to predict the performance of DPP pipelines. Since the model's goal is to predict the performance of DPP pipeline combinations, this prior knowledge must contain information on the effect of many DPP pipelines on the performance of different ML tasks. Thus, many datasets that can be used to train ML algorithms are needed, as well as an extensive set of DPP pipelines. To obtain this information, the project's first significant step is the data collection phase and, more specifically, constructing the knowledge base. This section will be detailed in Chapter 3.1. The knowledge base is obtained by collecting a series of ML model evaluations after transforming different datasets using various DPP pipelines. Afterwards, the raw model evaluation scores, contained within the knowledge base, are transformed into rank scores to help mitigate the effect of DPP pipeline performance scaling issues resulting from dataset difficulty and target scaling. This process results in a ranking table described in Chapter 3.2. From here, the information from the ranking table can be extracted to begin the meta-dataset creation phase. Meta-datasets train the meta-model and, like usual ML datasets, consist of inputs and outputs. The inputs, in this case, consist of meta-features extracted from the different datasets and algorithms whereby each instance in the input data describes a single *usecase* (i.e. a pairing of a dataset and an ML algorithm). Since the meta-model will learn a multi-output regression task, more than one output is used. This means that multiple meta-targets are used instead of a single target variable. Here, the different targets represent the performance of specific DPP candidate pipelines, comprising a DPP pipeline candidate pool, over the different usecases. The pipeline candidates are chosen based on several meta-target selection methods. Chapter 3.2.2 provides an overview of the different methods used. After the construction of the meta-dataset has been completed, the training phase begins. First, cross-validation is used to train the meta-model and determine ideal meta-model parameters, including model type, DPP pipeline transformations for the meta-dataset and model hyperparameters.

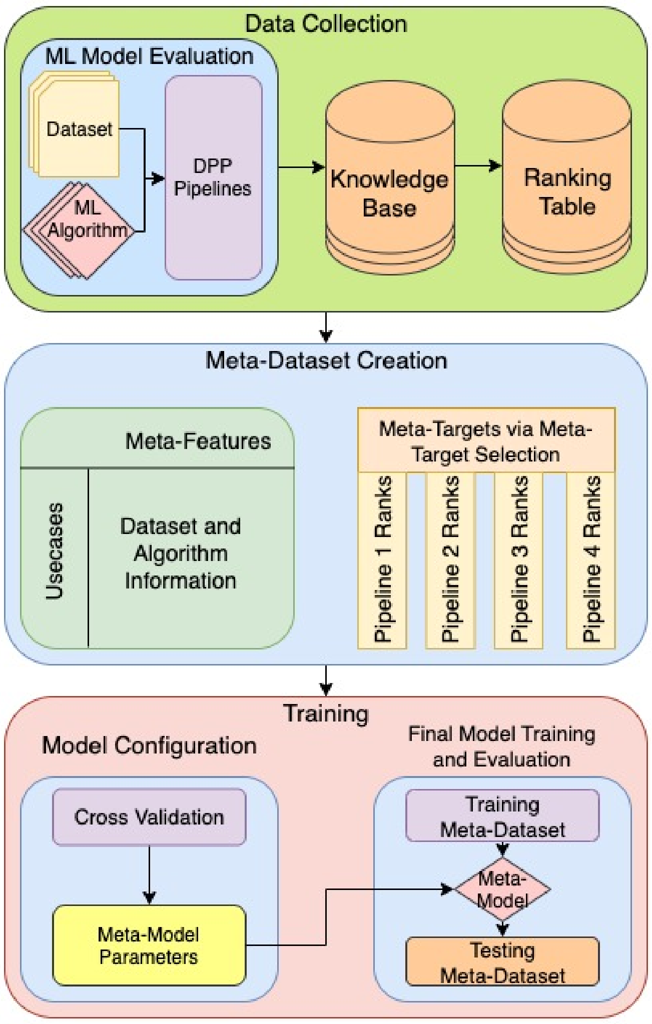


Figure 7: Overview of the important steps within the project to obtain a DPP pipeline rank prediction meta-model.

These model settings will then train the final meta-model to predict DPP pipeline performance. A testing set, constructed using the same methods as the training meta-dataset, is used to evaluate the final meta-model performance.

3.1 Knowledge Base

The project aims to create a meta-model to produce DPP pipeline performance predictions for any incoming (single target) classification or regression dataset. This tool is intended to be employed in the domain of production datasets; however, the scope can be increased further by extending the current knowledge base. Since learning a classification task is fundamentally different from regression tasks, especially with respect to the target variables and evaluations (see Table 1), two knowledge bases $\mathbf{K}$ will be collected separately $\mathbf{K}^t, t \in \{CL, RE\}$. A *knowledge base* can be defined as a collection of evaluations $K^t_{i,j} = K^t(u^t_i, \theta_j) \in \mathbf{K}^t$, whereby each evaluation is a result of the outcome obtained from the training of usecase u^t_i, and a single DPP pipeline combination $\theta_j \in \Theta, j = 1, ..., J$. A usecase consists of an ML algorithm $a^t_m \in A, m = 1, ..., M^t$ and ML dataset $d^t_n \in D, n = 1, ..., N^t$ pairing $u^t_i = u(d^t_n, a^t_m) \in U, i = 1, ..., I^t$ where the total number of usecases within $\mathbf{K}^t$ is $N * M$. Since a knowledge base is used for classification and regression tasks, the usecases u^t are split into two groups depending on their machine learning task t.

Machine Learning Task	Target Values	Evaluation Type
Classification	Categorical Classes	Accuracy Measures
Regression	Continuous Values	Distance Measures

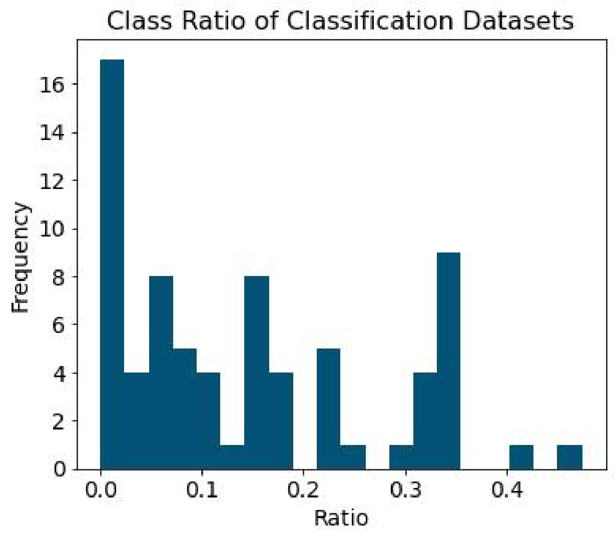
Table 1: Differences in targets and evaluation types between classification and regression tasks.

The following sections will explain the details regarding usecases and DPP pipeline combinations. After briefly describing the structure of a usecase, including information on the datasets and algorithms used in the knowledge base in Chapter 3.1.1, the construction process of the DPP pipeline combinations will be introduced. Subsequently, the algorithms used to run the collection of the model evaluations, as well as an overview of the collection results, will be presented in Chapter 3.1.3. Finally, a ranking method will be described, whereby the knowledge base raw scores are transformed into more manageable Scaled Ranks. These are introduced in Chapter 3.1.4 and allow for a better comparison of DPP pipelines without the direct influence of ML task and ML dataset difficulty.

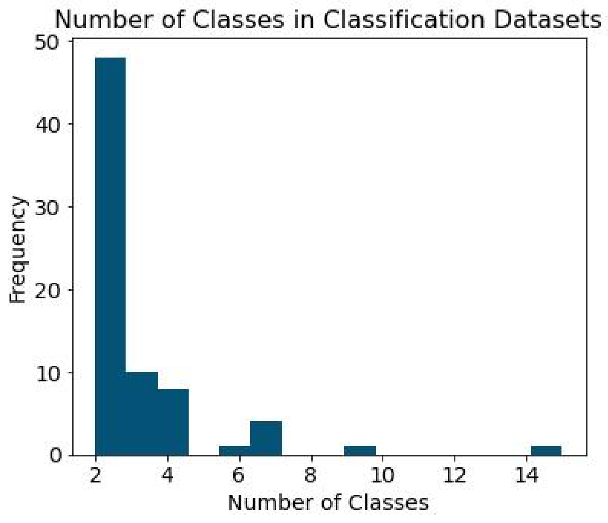
3.1.1 Usecases

As mentioned in the previous section, a usecase consists of the pairing of a dataset and an ML algorithm. The ML algorithm within a usecase is trained using the dataset previously transformed by each DPP pipeline. Combining an ML algorithm with a dataset avoids the concern of selecting an optimal model. Furthermore, if the resulting meta-model is used in practice, a developer can choose which ML model to employ for which DPP pipeline suggestions will be generated. Furthermore, if the trained meta-model is part of a more extensive AutoML-system that handles optimal model selection, this hypothetically suggested model could be used as an input to return DPP pipeline suggestions. Of course, to cater to a system that can recommend an ideal ML algorithm for a specific dataset, the set of ML algorithms included in the collection of usecases must be extensive. Otherwise, any predicted DPP pipeline performance would not be applicable should the intended algorithm be absent from the knowledge base $\mathbf{K}^t$.

Datasets The set of datasets D consisted of publicly available datasets collected by the Fraunhofer Institute of Production Technology (IPT). An overview of the datasets can be found in Appendix A. Many of these datasets were used in several other publications, including additional research based on meta-learning models (Laadan et al., 2019; Bilalli et al., 2018; Zagatti, 2021; Frye, 2023). Each dataset used was tabular and in the industrial and mechanical engineering domain. Since the Fraunhofer Institute IPT primarily works in a practical and industrial domain, industrial datasets were exclusively used instead of a collection of datasets from a broader range of domains. Thus, in most cases, the datasets contained features associated with sensory information during a mechanical process with the targets relating to fault and anomaly detection, performance and efficiency increases of mechanical processes or behaviour prediction of measures within aeronautical systems. Since the datasets were mainly production datasets that record measurements or other sensor recordings during production to identify faults or unusual activity, there was a high degree of class imbalance for classification datasets. Thus, the average minority class to majority class ratio was 0.203 in the classification datasets, and the average number of target classes was 2.95 (see Figure 8). These values indicate a strong need for resampling approaches in the recommended DPP pipelines. Besides the datasets being limited to production settings, diversity between the datasets in terms of size was present (see Table 2).



(a) Classification minority class ratio



(b) Classification class counts

Figure 8: General information on target classes for classification datasets.

By restricting the choice of datasets to those in the domain of industry and production, the eventual meta-model's ability to recommend DPP pipelines well is also restricted to be used only for datasets within the same domain. The knowledge base would need to be extended using a more varied collection of different datasets to obtain a model that can make DPP pipeline recommendations for any incoming tabular dataset. Despite this restriction, a meta-model's ability can still be assessed given that, when it comes to model evaluation, the testing data comes from the same domain. With the datasets selected and after some initial cleaning and the splitting of several datasets, a total of 28 classification and 18 regression datasets were available for the knowledge base.

An augmentation method was employed on compatible datasets (15 classification and seven regression sets) to increase the number of datasets. The augmentation consisted of two methods that each aimed to change the dataset such that new insights could be gained. The first method increased the number of missing values in a dataset by removing a random percentage of values between 0.04 and 0.06 from each column. The second method involved the conversion of numerical columns into categorical ones. This augmentation process was achieved by randomly selecting a set of columns and then discretising the numerical values into ten bins of equal frequency. Each numerical value within a given column was converted into a categorical value depending on its membership in one of the ten bins. For each augmented dataset, three different datasets were made, one increasing the number of missing values, the next increasing the number of categorical columns and the final applying both augmentation methods. After augmentation, the total number of datasets was 106, consisting of 73 classification and 33 regression datasets.

ML Task	Measure	Average	Std	Range
Classification	Num. Classes	2.959	2.071	13
	Num. Attributes	60.205	137.314	586
	Num. Instances	6710.027	10632.907	75960
Regression	Num. Attributes	23.061	63.89	372
	Num. Instances	6844.091	9597.026	41164

Table 2: Dataset descriptives for classification and regression datasets

ML Algorithms An extensive selection of ML algorithms $a_m^t \in A^t, m = 1, ..., M^t$ were used to collect the knowledge base evaluations from the `scikit-learn` python library. A list of all the classification and regression algorithms that were used can be found in Table 3.
Ideally, each learning algorithm would have used a set of different hyperparameters. This is because the effect of different hyperparameter settings can have vastly different impacts on not only the algorithm's performance but also on its behaviour (Lee & Giraud-Carrier, 2014). Thus, the results collected in the final knowledge base may not only differ based solely on algorithm performance, should the algorithm hyperparameters be tuned, but the effect of the DPP pipelines may also be largely different. However, since running each model with each hyperparameter configuration available would have significantly extended the time required to run the model evaluations, this loss of information was tolerated.
Though it was intended to run each of the 18 classification or regression algorithms for each dataset, there were instances where some algorithms could not be successfully run on specific datasets. These unsuccessful attempts mainly consisted of errors occurring somewhere along the DPP pipeline transformations that led to premature program terminations. Due to these unresolved errors, a decreased number of usecases, specifically for classification datasets. Overall the average number of ML algorithms for classification and regression datasets was 17.01 and 17.75, respectively. In total, the amount of all usecases $I^t = \sum_{n=1}^{N^t}(M_n)$ for classification amounted to $I^{CL} = 1242$ and for regression amounted to $I^{RE} = 586$, where N^t is the number of datasets for ML task t and M_n is the number of algorithms used on a particular dataset d_n^t.

3.1.2 DPP Pipeline Combinations

The DPP pipeline combinations $\theta_j \in \Theta, j = 1, ..., J$ of the knowledge base $\mathbf{K}^t$ refer to how a dataset will be transformed before training the ML algorithm and obtaining the evaluation score $K_{i,j} = K(u_i, \theta_j)$. The usecase $u_i \in U^t$ indicates which dataset and algorithm will be used. The set of all

Classification Algorithms		Regression Algorithms	
AdaBoost	Bagging	AdaBoost	Bagging
Decision Tree	ExtraTrees	Decision Tree	Elastic Net
Gaussian NB	Gaussian Process	Extra Trees	Gaussian Process
Gradient Boosting	Hist Gradient Boosting	Gradient Boosting	Hist Gradient Boosting
KNeighbors	LGBM	KNeighbors	LGBM
Logistic Regression	MLP	Linear Regression	MLP
Random Forest	Ridge	Random Forest	Ridge
SGD	SVM	SGD	SVM
Stacking	Voting	Stacking	Voting

Table 3: Classification and regression algorithms, taken from the scikit-learn python library, used in the knowledge base usecases. Note: the names of the models within the scikit-learn library include either the suffix Classifier or Regressor, depending on if they are regression or classification algorithms.

DPP methods used to create the different DPP pipelines consisted of methods spanning the DPP categories mentioned in Chapter 2.2. A list of the DPP methods and their membership to the categories can be found in Table 4. Most methods were taken from online python libraries, mainly including the `scikit-learn` python library and the `category encoders` library (McGinnis & Westenthanner, 2015). The Z-Score Substitution, Random Noise Addition, and Remove Correlated Features methods, however, were implemented specifically for the project (see Appendix D for pseudo code of these methods, which were implemented by Frye (2023)).

DPP pipelines were constructed employing an exhaustive iterative procedure, iterating through each DPP task category (see Table 4) and selecting one method from within that category. Also included in a group of DPP methods was an option to choose a Dummy Method that would not transform the dataset. With the list of possible DPP methods that can be combined, a factorial process was carried out for each DPP task. Since nine DPP tasks were present, the maximum length of a DPP pipeline was nine (i.e. Imputation, Encoding, Substitution, Noise Addition, Gaussian Transformation, Scaling, Feature Extraction, Feature Selection and Over and Undersampling). Furthermore, as the Dummy Method was included as an option within each DPP task, some DPP pipelines did not transform the dataset in any way. These baseline pipelines allowed for comparing model performance before and after preprocessing. In many cases, however, this exact comparison was not possible since running a large portion of datasets without applying any DPP methods would not run successfully. The datasets in question would be those with missing values or non-numerical categorical columns. Thus, in datasets where imputation methods or encoding methods were necessary to be able to run, no true baseline results could be obtained. Instead, the minimum number of DPP method transformations was used as baselines. Scores were obtained after transforming the datasets using either encoding, imputation, or both, depending on which were required.

The order in which the DPP methods occurred within a pipeline was fixed. For example, in a pipeline with a maximum of nine steps, the first methods would refer to those deemed mandatory, including imputation and encoding steps. Subsequently, transformation methods (substitution, transformations and scaling) followed. After these methods, the dimensionality of the data would be reduced using feature extraction and feature selection methods. Finally, random noise would be added to the model, and the data would be resampled. A fixed order of possible DPP methods was required as the number of possible DPP pipeline combinations had to be kept manageable, with the maximum number of DPP method combinations in the current configuration being $I_{max} = 15360$. If, on the other hand, the different method types could occur in any order, this number would be taken to the power of the number of method types reaching a total combined count of $4.7 * 10^{37}$. This amount of methods are infeasible and unnecessary. For one, many transformations require that no missing and non-numerical values are contained within a dataset. Thus the mandatory steps were required to be at the beginning of pipelines. Furthermore, as highlighted by Giovanelli et al. (2021), the order in which some methods follow others can lead to better results. For instance, data reduction methods are best suited to appear after data transformations since better reduction occurs when the values have been optimally scaled. In the case of resampling methods, the best results can be expected when the newly created data, obtained by employing these methods, were generated based on scaled and transformed data (Bagui & Li, 2021).

Category	DPP Task	DPP Method
Data Cleaning	Imputation	Median Imputer
		Mean Imputer
		Last Observed Imputer
	Encoding	One-Hot Encoder
		Target Encoder
		Ordinal Encoder
Data Transformation	Substitution	ZScore Substitution
	Noise	Random Noise
	Gaussian Transformer	Power Transformer
	Scaling	Standard Scaler
		Min-Max Scaler
Data Reduction	Feature Extraction	PCA
	Feature Selection	Recursive Feature Elimination
		Variance Threshold
		Remove Correlated
Data Resampling	Undersampling/Oversampling	Random Undersampling
		SMOTE
		SMOTEENN
		SMOGN

Table 4: Overview of all DPP methods in their respective category. Implementations from `scikit-learn` were used for most of the methods. On the other hand, the encoders were sourced from the `category encoders` library. For the Remove Correlated method and ZScore Substitution method, personally developed implementations were created

DPP pipelines were created for each usecase in the knowledge base. As mentioned, the maximum number of possible DPP pipeline configurations was $I_{max} = 15360$. However, the number of combinations eventually used to transform the usecases was much lower. The reason for this lower pipeline count can be traced to several factors. For one, only datasets that contained missing values were given DPP pipelines that included imputation steps, and only those with categorical features had pipelines that included encoding steps. Naturally, if a dataset contained either no missing values or categorical features, the number of pipelines would decrease by a factor of 3 (or 3^2 if a dataset had no missing values and no categorical features) since each pipeline included neither encoding nor imputations steps. A further reason for the lower DPP pipeline count was a set of rules applied to constructing DPP pipelines. These rules aimed to reduce the number of errors caused by DPP pipelines during the data collection. After applying these rules, PCA and Variance Threshold steps were only allowed if the pipeline included a power transformer, allowing for better scaling of the data before these reduction methods. This helped with the performance of these two DPP methods and avoided certain erroneous infinite value transformations. Finally, since classification datasets only benefited from resampling methods, many combinations were left out for regression datasets. On top of the systematic reasons for the reduction of DPP pipeline combinations, a small percentage of different DPP pipelines were caused by errors that hindered the production of a performance evaluation for those specific pipelines on a certain usecase. For an overview of the actual numbers of DPP pipelines successfully evaluated for classification and regression, usecases, see Table 5.

ML Task	Imputation Required	Encoding Required	Both Required	No Cleaning Required
Classification	1738.92	1719.6	5014.44	573.43
Regression	654.23	515.1	1581.75	169.31

Table 5: Average number of different DPP pipelines that a single regression or classification usecase was successfully transformed by depending on the level of data cleaning required

3.1.3 Collection

To obtain the performance evaluations $K_{i,j} = K(u_i, \theta_j)$ that make up the knowledge bases $\mathbf{K}^t, t \in \{CL, RE\}$, the same algorithm designed by Frye (2023) was employed to collect each model performance after DPP pipelines were applied to the usecase datasets. The process was carried out for both regression usecases and classification usecases.

Procedure The collection algorithm automated several steps to amass model evaluations for different ML algorithms on a dataset. First, before training any models or using DPP pipelines, necessary transformations were applied to the datasets. The necessary transformations consist of a handful of modifications that benefit each dataset and thus were not included in the DPP pipelines. They handle the removal of rows and columns that will not contribute to the model's success, such as constant and duplicate rows and columns. Next, the list of all possible DPP methods was created, depending on whether or not the dataset included missing values and categorical features and if the dataset was a regression or a classification dataset. With the list of all applicable DPP pipelines to be used on a dataset formed, the data could be split into a training and testing set. This was done before the application of DPP methods to avoid data leakage. The train-test split separated the data into a stratified sample (in the case of a classification dataset) of 70% training data and 30% testing data. The training data was then used to fit the DPP methods, which subsequently transformed the testing data.
Rather than running each DPP method in a pipeline from start to finish, some transformations were only allowed to affect parts of the datasets. The reason for this lies with the categorical encoders. The

methods within a pipeline follow a particular order as mentioned in Chapter 3.1.2, whereby if both dataset encoding and dataset transformation steps are in the pipeline, the encoding will always happen first. The problem with this stems from the behaviour of encoding methods. The resulting transformation after encoders, such as binary and especially One-Hot encoders, can potentially leave the dataset with a vastly inflated number of columns. As mentioned in Chapter 2.2.2, a One-Hot encoder will add columns for each of the different instances within a categorical column. Thus, encoders, besides Target encoders, have the potential to litter the dataset with large numbers of sparse columns. Although sparse columns are known to benefit the knowledge acquisition of ML algorithms, feature reduction steps such as PCA would remove them. This issue was avoided by removing the categorical columns generated using encoding methods and returning them after the subsequent transformation and reduction methods had been run on the dataset.
After successfully transforming a dataset using a DPP pipeline, each of the corresponding ML algorithms was trained using default parameters on the training set, resulting in a performance evaluation of that model. For example, if the dataset required a classification algorithm, the F1-Score was calculated. Conversely, the Root Mean Squared Error (RMSE) was calculated for regression datasets.

Outcomes The resulting model evaluation obtained from running the collection algorithm over each dataset shows that the DPP pipelines can cause significant differences in the performance of a model. The example of the usecase using the dataset ”Steel Plates Faults both”, and an SVM classifier can be taken to illustrate this effect (see Figure 9).

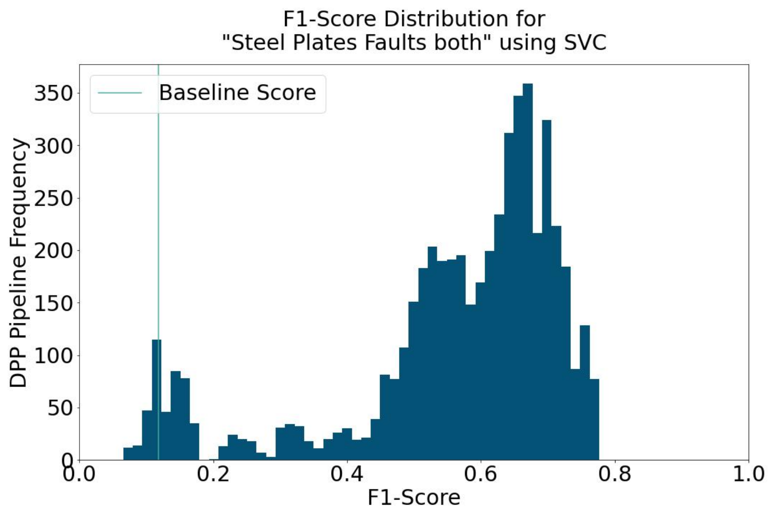
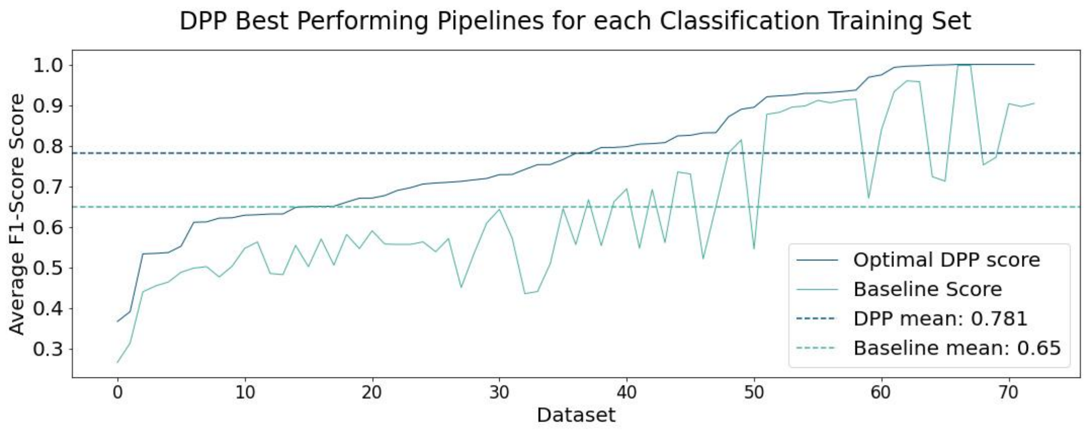


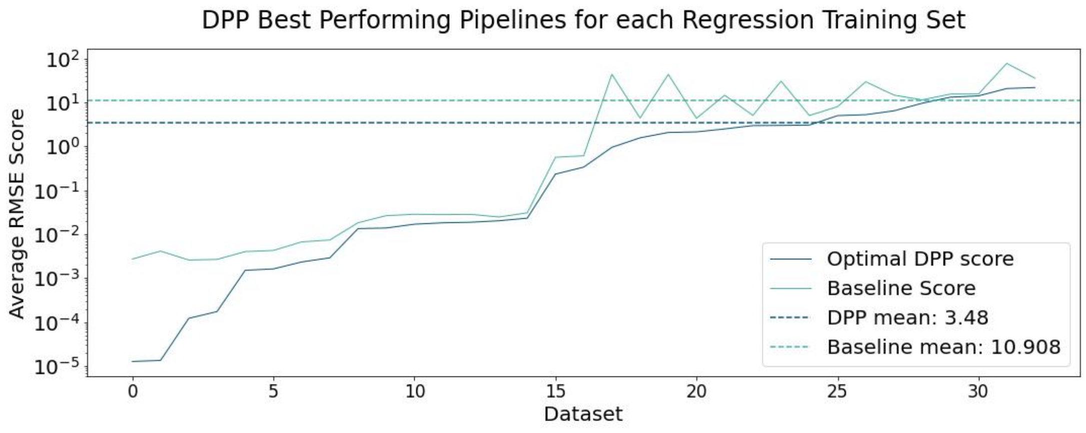
Figure 9: F1-Score distribution of a Steel Plates Faults both datasets using SVC.

In this particular usecase, the range of F1-Scores equates to 0.71, with the best-performing pipeline helping the model to achieve an F1-Score of 0.77 and the worst pipeline leading to an F1-Score of 0.06. Furthermore, these results show that using DPP pipelines can greatly improve the performance of ML algorithms since the baseline pipeline, in this case, an encoder and an imputer, only led to an F1-Score of 0.12. This effect was found for almost all usecases, whereby the best-performing pipeline for a given usecase had a higher performance than the baseline scores. When taking the average scores achieved on each classification dataset over all of the ML algorithms into account, we can see that the average best DPP pipeline performance achieved an F1-Score of 0.78. In contrast, the average baseline F1-Score is 0.65 (see Figure 10), indicating that, just by optimising the DPP pipelines for a given dataset, one can, on average, increase the F1-Score by 0.13.

The calculation of the regression scores is more complex to compare since only the RMSE was collected in the knowledge base. Furthermore, given that the RMSE value depends on the scale of the target values used in the dataset, datasets that naturally contain target values on a larger scale will also have distance-measure outcomes on larger scales. The values of the optimal scores across datasets illustrate the effect of the different scales. The smallest optimal RMSE score that was recorded was $\text{argmin}_i(\text{argmin}_j RMSE_{i,j}) = 1.26 * 10^{-5}$ and the largest optimal RMSE measure being $\text{argmax}_i(\text{argmin}_j RMSE_{i,j}) = 21.81$, both the median and mean values indicate that best DPP pipelines outperform the baseline model evaluations. Here, the mean difference was 7.42 RMSE in favour of DPP methods. Unfortunately, since collecting the model evaluations for each DPP pipeline combination within the knowledge base took a substantial amount of time, each evaluation was collected only once. Thus, no tests of statistical significance made conducted. Following this, no claims can be made about the significance of improvements observed in the models when using DPP pipelines compared to not using DPP pipelines. However, these results prop up the already established finding that using DPP pipelines will improve model performances.



(a) Classification Datasets



(b) Regression Datasets

Figure 10: Affects of applying DPP pipelines to each dataset (numbers associated with datasets can be found in Appendix E). Optimal DPP Score refers to the averaged F1-Score or RMSE score achieved by the best-performing pipeline over each usecase using the same dataset. The Baseline score refers to the score achieved when not performing any or only the strictly necessary DPP methods (categorical encoding or missing value imputation).

Though difficult to compare the performance of ML algorithms on different regression datasets with one another, the internal order of the distance measures within a usecase allows for the ideal pipelines to be identified and used for learning. The subsequent section will document how the information within the knowledge bases was transformed, which allowed for the comparison of DPP pipelines between datasets.

3.1.4 Ranking

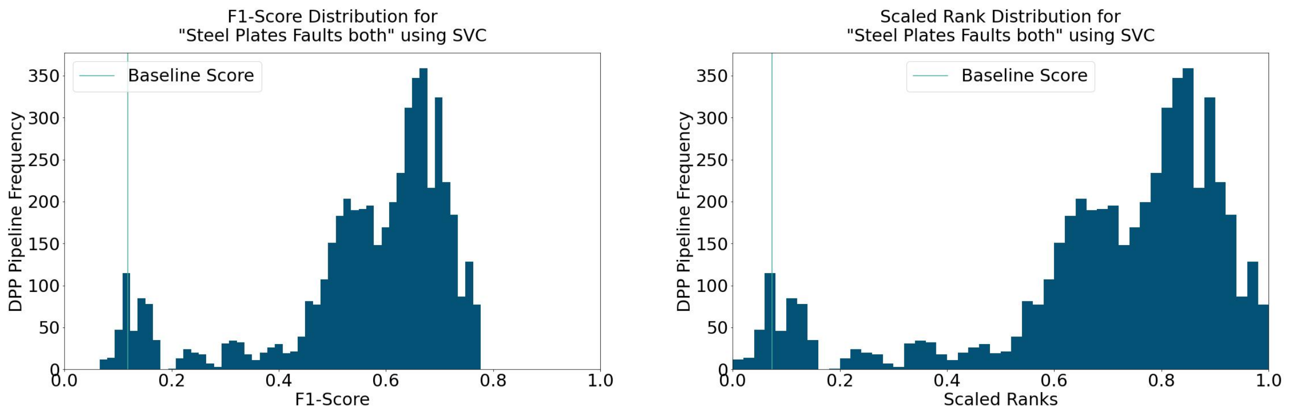
The final step in the knowledge base collection procedure was ranking the ML model evaluation results within the knowledge base. So far, the knowledge bases $\mathbf{K}^t, t \in \{CL, RE\}$ consist of raw model evaluations depending on a usecase and a DPP pipeline configuration $K_{i,j} = K(u_i, \theta_j)$. Though these raw scores represent the actual scores that were achieved, there are several reasons why transforming them into a more convenient structure is advantageous. For one, the performance measures are inverted in the regression and classification knowledge bases. Where a large value indicates a well-performing model for the classification knowledge base, a small value is ideal for regression. More importantly, as previously noted, the RMSE scores in the K^{RE} depend on the overall scale of the values within the dataset, making a comparison between regression model evaluation impossible. Though the scale of the F1-Scores is the same for the outcomes over each usecase, classification datasets have a similar problem as some datasets are easier than others for ML algorithms to solve. Thus, a top-performing pipeline that helps an ML algorithm achieve an acceptable accuracy score for a challenging dataset will become overshadowed by the accuracy scores attributed to any DPP pipeline that transforms easier datasets. These issues were rectified by using a transformation technique used by (Frye, 2023), whereby the model evaluations within the knowledge bases are transformed into ranking tables $\mathbf{R}^t, t \in \{CL, RE\}$.
The ranking was carried out by ranking the performance measures of each DPP pipeline within a usecase $R_{i,j} = R(u_i, \theta_j)$ using a Scaled Distance Ranking method.

$$ScaledDistanceRank = \frac{K(u_i, \theta_j) - min(K(u_i, \theta_{all}))}{max(K(u_i, \theta_{all}) - min(K(u_i, \theta_{all})))} \tag{5}$$

This method negates the problems stemming from the differing inter-dataset scales and difficulty levels since the Scaled Ranks were not only bound between the same limits, but the maximal and minimal Scaled Rank scores were now identical between datasets. Following this, the pipeline that scored the optimal score (either the highest F1-Score or lowest RMSE score) would be given a Scaled Rank score of 1, and the least optimal score would be set to 0. Besides solving the scaling issues, using the Scaled Rank to rank the usecase pipelines also allowed the shape of the distribution of the usecase model evaluations to be maintained (see Figure 11). Thus, the ranking procedure can be seen as a mapping from the knowledge base to the ranking tables $\mathbf{K}^t \mapsto \mathbf{R}^t$.

3.2 Meta-Dataset Creation

To be able to train a meta-learning model, the information contained in the knowledge bases must be organised into meta-datasets $d^t_{meta}, t \in \{CL, RE\}$ so that a meta-model can learn to predict DPP pipeline performances. Like regular datasets that train ML models, the meta-datasets consist of inputs and outputs. The inputs, in this case, describe usecases and are constructed by extracting meta-features from the individual datasets, which, in turn, describe datasets and algorithm characteristics. The outputs or meta-targets contain information about DPP pipeline performance (see Figure 12). As previously mentioned, the meta-model used in this project will be a multi-output regression model



(a) F1-Scores (b) Scaled Ranks

Figure 11: F1-Score (a) and Scaled Rank (b) distribution of a Steel Plates Faults both dataset using SVC. It can be seen that the shape of the distribution remains the same after the transformation to Scaled Rank scores.

predicting the values associated with several meta-targets. Thus, a smaller set of DPP pipelines will be chosen as candidates for regression and classification datasets to avoid predicting the performance of all DPP pipeline combinations.

In the following sections, the construction of the meta-datasets will be presented. First, the method of obtaining input values via meta-feature extraction is described in Chapter 3.2.1, whereby each instance of the meta-dataset inputs characterises a usecase in terms of dataset and ML algorithm used. Subsequently, the creation of the meta-dataset outputs via meta-target selection is described in Chapter 3.2.2. Initially, the notions of reliable and diverse meta-target selection are introduced, followed by a description of the novel and commonly used meta-target selection methods. In addition, the DPP pipelines, chosen by the different meta-target selectors, are described. Finally, the datasets used to test the final meta-model are introduced in Chapter 3.2.3.

3.2.1 Meta-Feature Extraction

The first step towards constructing the meta dataset is to obtain the input values (see inputs in Figure 12). Then, as mentioned in Chapter 2.3.2, meta-features are used to describe a task numerically, helping to identify similarities between tasks. By identifying task similarities, the meta-model can learn underlying patterns representing the effect of DPP pipelines on the Scaled Rank scores. The meta-features aim to characterise the dataset of a particular usecase. They range from simple handcrafted meta-features, such as the number of missing values, instances and attributes of a dataset, to complex ones for which the python library `pymfe` was used. A list of the handcrafted meta-features can be found in Table 6. Additionally, a description of the 89 meta-features used from external libraries can be found in Appendix B.

With the two sets of meta-features (handcrafted and `pymfe` meta-features), the overall number of columns in the meta-dataset d^t_{meta} was 100 for the classification dataset and 103 for regression datasets. The difference in the number of meta-features between classification and regression sets stems from the nature of some meta-features. For example, those describing classes, such as *Number of Classes* and *Class Distribution*, were specific for classification usecases. In contrast, the meta-features: *Target Skewness*,*Target Kurtosis*, and *Number of Target Outliers* were specific to regression usecases. Since some of these meta-features are categorical, the datasets required mandatory encoding, which

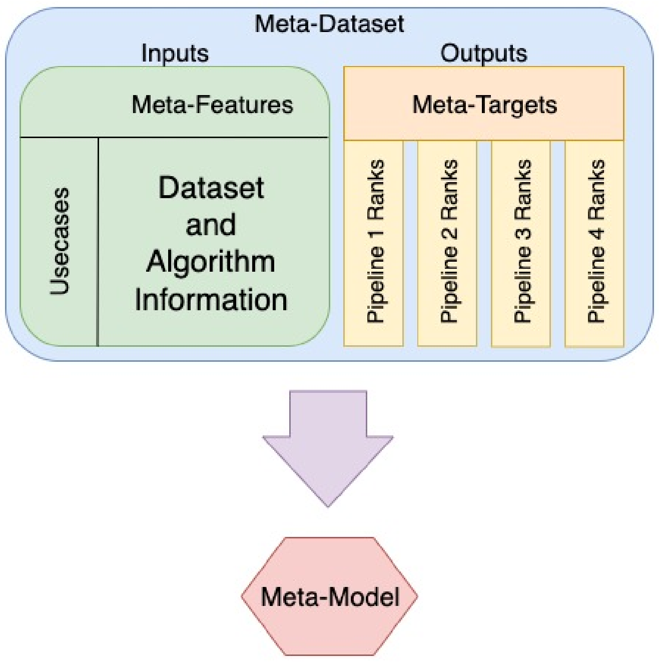


Figure 12: An overview of the structure of the meta-dataset. Each instance of the dataset (usecase) consists of meta-feature information describing the dataset and algorithm characteristics. For each usecase, the Scaled Rank scores of N DPP pipelines, in this case four, are used as training targets. This meta-dataset is used to train the resulting meta-model.

was accomplished using a binary encoder. A binary encoder was chosen over a One-Hot encoder to keep the dimensionality increase as low as possible. Since the ML algorithms that were eventually chosen to become the meta-model were not neural networks (see Chapter 3.3.2), the usual decrease in performance usually observed when using binary encoding was preferred over the explosion in dimensionality.

An alternative would have been using a Target encoder with the average meta-target scores for each category as an encoding, as this method does not increase the number of columns within a dataset. However, this was not implemented since the Target encoder from the `categorical encoders` library does not support multi-output encoding. After encoding, the respective dataset dimensions were 1142x132 for classification usecases and 1142x130 for regression usecases.

The attribute-to-instance ratio is quite large, which can cause problems during training due to the curse of dimensionality. Therefore, data reduction DPP methods were used on each meta-dataset to counteract this issue. This was accomplished via the PCA feature reduction method. PCA was employed since it is a commonly used method for continuous features, combining many features into principal components rather than removing entire features. Additionally, since only DPP methods contained within the selection of DPP pipelines used as meta-targets are to be considered, more involved methods, such as tSNE reduction, were excluded. Furthermore, a PCA variance analysis can determine the ideal number of principal components to be included in the final dataset. The results of the variance analysis run on the numeric columns of the datasets show that approximately 99% of the variance is achieved by only using 25 principal components for both classification and regression datasets. Thus, the datasets were reduced to 25 principal components plus the encoded categorical columns before training began.

Universal Meta-Features	Classification Specific	Regression Specific
Number of attributes	Number of Classes	Target Skewness
Number of Instances	Class Distribution	Target Kurtosis
Number of Numeric Features		Number of Target Outliers
Number of Categorical Features		
Number of Missing Values		
Number of Outliers Zscore		
Number of Outliers Boxplot		

Table 6: Overview of the handcrafted meta-features directed towards datasets and algorithms

3.2.2 Meta-Target Selection

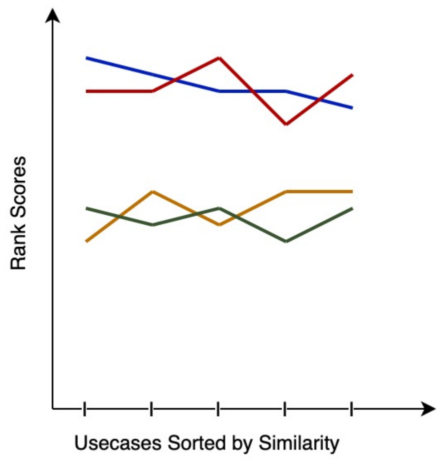
The output or meta-target is the item the meta-model is trained to predict. Depending on the task the meta-model has been given, the targets may consist of different classes c belonging to a set $c \in C$. Usually, the size of C is kept relatively small since increasing the size of classes to be predicted, in turn, increases the difficulty of the task. This increase in task difficulty is a major problem when recommending the best DPP pipelines for a dataset. Therefore, research involving DPP pipeline recommendation has consistently resorted to drastically reducing the set of pipelines instead of predicting the best possible pipeline from the entire set.

Unlike the PRESISTANT meta-model, designed by Bilalli et al. (2018), which uses a rule-based approach to determine which DPP pipelines would be used to transform incoming datasets, this shortened set C^* of pipeline candidates usually consists of well-performing DPP pipelines. Often, the candidate pipelines are chosen based on a DPP pipeline ranking obtained from knowledge bases (Laadan et al., 2019; Zagatti, 2021; Frye, 2023). Thus, the choice of which DPP pipelines can be predicted depends on a statistical approach.

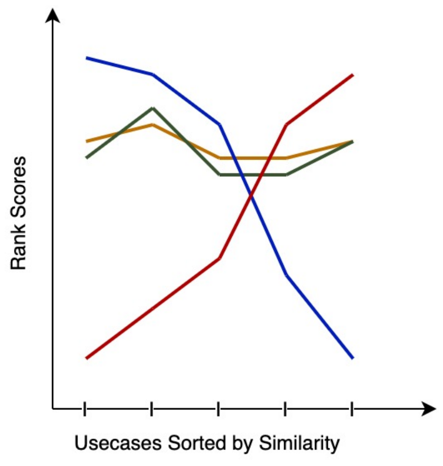
Given that little has been written about different meta-target selection methods, the current project aims to shed light on the effect of using different approaches. Furthermore, since the current model will predict the Scaled Rank score $R(u_i, \theta_j^*)$ of several DPP pipelines $\theta_j^* \in \Theta^*, j = 1, ..., J^*$, investigating different methods for meta-target selection will not only help further the information about meta-learning but also aid in achieving better performance for the current meta-learner. Given that the current project aims to use a knowledge base, a rule-based approach was not considered. Instead, methods were designed based on the performance information of DPP pipelines with different use-cases. With this in mind, the first step in defining a new pipeline selection method was to define what desirable characteristics a DPP pipeline should have to be selected to be a member of the candidate pool.

Reliability vs Diversity The Meta-Targets chosen in previous DPP pipeline meta-learning studies were predominantly based on average performances. In most cases, the top N pipelines were selected after being ranked according to achieved evaluation metrics attributed to DPP pipeline transformations (Laadan et al., 2019; Zagatti, 2021). An alternative method was employed by Frye (2023). Here DPP pipelines were chosen by identifying DPP pipelines that appeared in the most top N ranked pipelines of usecase subgroups. In this study, the groups were determined by the algorithm used.

Nevertheless, these approaches limit the candidate pool to pipelines that consistently produce favourable results for all tasks within a knowledge base. The objective is thus to select pipelines with high *reliability* and is illustrated theoretically in Figure 13a. Here we can see an example of four pipelines and their performances after transforming the datasets within the usecases represented along the x-axis. The scores achieved by the red and the blue pipeline both obtain solid results in all usecases, whereas the scores achieved by the orange and green pipelines are weak. Therefore, the desired choice would be to include the blue and the red pipeline in the candidate pool but exclude the orange and green.



(a) Reliability



(b) Diversity

Figure 13: The objectives for selecting pipelines based on reliability (a) and diversity (b) are illustrated in the two hypothetical plots. The lines represent Scaled Rank scores achieved by a specific pipeline of a given usecase, the different colours represent different pipelines, the y-axis represents the achieved Scaled Rank score, and the x-axis represents different usecases sorted by similarity. Thus, the usecase on the far left is the least related to the usecase on the far right. In both cases, the better choice of the pipeline would be those producing the red or blue data points. In *a*, the red and blue are shown to achieve better Scaled Rank scores over the different usecases than the orange pipeline that consistently returns low scores. In *b*, the red and blue pipelines are the ideal choice for specific usecases and would make selecting the better pipeline according to usecase similarity easier. On the other hand, the orange pipeline's performance appears not to be well related to usecase similarity and would make learning difficult.

When the meta-learner aims to recommend the ideal pipelines for a usecase, this pipeline selection method is helpful to ensure that, no matter which pipeline is predicted, it will likely increase prediction performance. However, regarding the model's ability to distinguish which pipeline is the ideal choice for a specific usecase, the meta-learner must be very precise to successfully distinguish one DPP pipeline from the next since the difference between pipelines is small. Chosen pipelines based on high reliability will thus lead to lower meta-model accuracies if the model's margin of error is considerable. Furthermore, selecting DPP pipelines based solely on average performance increases the danger of crowding the candidate pool with well-performing, homogeneous pipelines. For example, should the distribution of the usecases be biased toward a particular attribute, for instance, a need for scaling, then the top N pipelines may consist of DPP methods that only perform data transformations that focus on scaling. For this reason, a new pipeline selection objective is proposed.
An alternative to choosing reliable meta-targets that will perform similarly for many usecases is to choose meta-targets that perform well for specific ones. This pipeline selection objective is to select

pipelines with high *diversity*, illustrated in Figure 13b. Here the blue and the red pipelines achieve strong or even optimal performances in only specific usecases and are not well suited for others. With the optimal pipeline being more distinguishable than reliable pipelines, the task of the meta-learner is more straightforward. On the other hand, the difference between the orange and green pipeline results in Scaled Rank scores that are similar across the usecases and makes the task of the meta-learner more difficult. Furthermore, since the orange and green pipelines were chosen based on high reliability, their average score over all usecases may be high; however, specialised pipelines may outperform them in the ideal usecases. The idea behind diverse pipelines is to select heterogeneous pipelines and focus on specific problems within datasets to be optimally processed. This paper will give no precise definition of reliability and diversity, and the above explanation and that given in Figure 13 is only a suggestive take on the issue. However, two intuitive approaches for the different pipeline selection methods were designed to highlight the differences between the meta-target objectives. The aim is to test whether using a diverse selection method can lead to increased meta-learning accuracy. The first method refers to grouping methods, which aim to select pipelines with high diversity. The second method follows a simplified approach that targets reliable pipelines. Both pipeline selection types will be explained in the following sections.

Grouping Methods The first pipeline selection type aims to identify pipelines that perform well for different subsets of usecases and thus find pipelines with high diversity. The subsets are defined by grouping usecases and finding the best-performing pipelines for each selected group. Given that the algorithm type is the most important feature in determining ideal DPP pipelines, according to Frye (2023), the usecases were grouped according to the different classification or regression ML algorithms $a_m^t \in A^t, m = 1,...,M, t \in \{CL, RE\}$. In total, three grouping methods were designed for this project $\mathbf{G}^r, r \in \{IAG, SAG, RAG\}$, each following a different strategy to combine usecases $u_i \in U, i = 1,...,I$ according to their ML algorithms. The resulting groups obtained after employing a grouping method G_n^r were thus a subset of all usecases $G_n^r = G(A_n^{t*} \subseteq A^t) \subset U, n = 1,...,N <= M$.
The first grouping method was adapted from a similar approach taken by Frye (2023). This method, the *Individual Algorithm Grouper* (IAG) $\mathbf{G}^{IAG}$ grouped the usecases by each algorithm individually $G_n \in \mathbf{G}^{IAG}, n = 1,...,N = M$, meaning there were as many groups as there were ML algorithms. Although the IAG method was adapted from Frye (2023), the steps taken in this project are aimed at selecting diverse DPP pipelines rather than following a reliable approach. The *Similar Algorithm Grouper* (SAG) $\mathbf{G}^{SAG}$, on the other hand, combined algorithms into subgroups according to similar algorithm characteristics. These characteristics were based on the algorithms' functionalities, such as training methods and structures. Table 7 shows the algorithm subgroups used in this grouping method. Finally, the *Random Algorithm Grouper* (RAG) $\mathbf{G}^{RAG}$ grouped algorithms randomly into five similar-sized groups. This method was included to highlight the importance of usecase similarity when aiming to select pipelines with high diversity. It was expected that the pipelines chosen using the random grouping method would show similar scores across the five groups, resulting in pipelines with high reliability rather than a high diversity.
During the selection process, the best-performing pipelines within a group were identified by extracting the usecases from the ranking tables $\mathbf{R}_n^t = R(G_n^r, \Theta)$ where $u_i \in G_n^r, i = 1,...,I_n$, which contained the Scaled Rank scores over all DPP pipelines for the usecases specified by each algorithm G_n^r. From here, the median scores over the usecases within a group were calculated $median(\mathbf{R}_n^t(G_n^r, \theta_j)), j = 1,...,J$. This resulted in a collection $\mathbf{R}_{Ave}^t$ of median scores that each pipeline achieved for usecases defined by the different groups $\mathbf{R}_{n,j}^{t,Ave} = median(R(G_n^r, \theta_j)) \in \mathbf{R}^{t,Ave}, n = 1,...,N, j = 1,...,J$. The median scores were taken instead of average scores to reduce the impact of outliers. Five subsequent steps were used to obtain pipelines that were likely to be diverse:

Group	Algorithm
Ensemble-Based Algorithms	Voting C/R
	Gradient Boosting C/R
	AdaBoost C/R
	Bagging C/R
	Stacking C/R
	Histogram Gradient Boosting C/R
	LGBM C/R
Linear Algorithms / SVM	SGD C/R
	Ridge C/R
	Logistic Regression
	Elastic Net
	Linear Regression
	SVM C/R
Bayes Algorithms	Gaussian Process C/R
	Gaussian Naive Bayes
Tree-Based Algorithms	Random Forest C/R
	Decision Tree C/R
Clustering Methods	KNN C/R
Neural Networks	MLP C/R

Table 7: Groups of algorithms used in the algorithm groups pipeline selection method. Note: C/R stands for Classifier/Regressor. For these algorithms, the implementation used for both classification and regression datasets were included. Additionally, SVM was included in the linear group since the prediction is linear after the kernel trick has been performed and if a linear kernel is used.

- **Max Scores**
  Only pipelines that perform exceptionally well for specific usecases are wanted in the candidate pool. Thus the total number of pipelines in the search space was reduced. Pipelines $\theta_j \in \Theta$ were removed whose maximum group performance of a pipeline $max(\mathbf{R}_j^{t,Ave}$ did not appear in the 90th percentile of scores for the group of the pipelines maximum performance. This resulted in a reduced set of possible pipelines to be considered $\mathbf{R}_{n,k}^{t,Red} = mean(R(G_n^r, \theta_k)) \in \mathbf{R}^{t,Red}, n = 1, ..., N, k = 1, ..., K < J$.

- **Pipeline Group Ranks**
  The pipelines in the resulting candidate pool should perform well only in specific usecases rather than in all. Each pipeline's performance over the different groups was used to help select pipelines that perform well in certain groups instead of others. This information was obtained by transforming the median pipeline group scores into pipeline group rankings by ranking the scores achieved by each pipeline according to the achieved scores for each of the groups $\mathbf{R}_{n,k}^{t,Rank} = rank(\mathbf{R}_{N,k}^{t,Red}) \in \mathbf{R}^{t,Rank}$. Thus for each pipeline, the group scores are replaced with rankings $\mathbf{R}_{n,k}^{t,Rank} \in \{1, ..., N\}$ for each of the groups $G_n^r, n = 1, ..., N$

- **Highest Pipeline Rank Count**
  Since there are usually more groups than selected pipelines, a single pipeline that will perform well for only one usecase group cannot be chosen. Instead, the grouping pipeline selection method aims to search for pipelines that show top performances over multiple groups, narrowing the search to candidates that perform well for as many usecase groups as possible. Thus, the number of pipelines in the reduced ranking table $\mathbf{R}^{t,Red}$ were further reduced to pipelines that achieved high rank scores in multiple groups. This second reduction resulted in a ranking table of potent pipelines $\mathbf{R}^{t,Pot}_{n,l} \in \mathbf{R}^{t,Pot} \subset \mathbf{R}^{t,Red}, l, ..., L < K$.

- **Group Limiting**
  The number of total pipelines was again limited by choosing the top three pipelines per group based on the maximum score achieved in each group to avoid unbalanced amounts of potent pipelines, which may occur from some groups leading to better results than others. This step was the final reduction performed on the list of pipelines $\mathbf{R}^{t,Limited}_{n,o} \in \mathbf{R}^{t,Limited} \subset \mathbf{R}^{t,Pot}), o, ..., 3 * N$ and helped to ensure that remaining pipelines are among the best in the list of pipelines for the different groups. Furthermore, this step reduced the search space for the following steps in the pipeline selection methods.

- **Potent Pipeline Combinations**
  With a selection of pipelines that perform well in the different groups and the goal of finding a candidate pool of pipelines that each have strengths in as many different groups as possible, an ideal combination of pipelines was needed. A grid search over all possible combinations of the pipelines within the limited ranking table $\mathbf{R}^{t,Limited}$ was then conducted to obtain this ideal combination. These combinations are potential candidate pipeline pools and consist of J^* pipelines. Throughout the search, the number of times pipelines within a particular combination achieved the highest group ranks is counted. Pipeline combinations that achieve the highest number of maximal ranks over the pipelines within the combination are collected for the final step.

- **Pipeline Candidate Pool Selection** With a selection of pipeline pools collected that achieved the highest possible total of top group ranks, the best pipeline candidate pools can now be chosen. This choice is based on each pipeline's overall highest average performance within the pipeline candidate pool.

This method of selecting a DPP pipeline candidate pool allowed for pipelines to be chosen that performed well in a subset of usecases while disregarding the performance achieved in others. This allowed for greater differences in pipeline performance across different usecases, simplifying the learning task that the eventual meta-model will attempt to solve.

Statistical Methods A second method for pipeline selection was used to compare the grouping method's motivation of searching for pipelines with high diversity. This *Statistical Pipeline Selector* (SPS) was much more straightforward. Instead of grouping the usecases into separate groups, the median Scaled Rank score was collected over all usecases U^t in the ranking tables $R^{t,ave}_j = median(\mathbf{R}^t(U^t, \theta_j))$. Then, the ranking tables were sorted according to the median values $R^{t,ave}_j$. Subsequently, the pipeline candidate pool was chosen $\theta^*_j \in \Theta^*, j = 1, ..., J^*$ by selecting the top J^* pipelines from the sorted DPP pipeline ranking table. Pipelines chosen using only statistics over all usecases allow for a more straightforward pipeline selection method but only look for pipeline *reliability*. Models will be trained using the pipelines obtained by both selection types, and the best approach will be used for the final model.

Chosen Pipelines Four pipelines were chosen to be in the resulting pipeline candidate pools ($J^* = 4$). These pipeline sets differed for the four pipeline selection methods $PS^r, r \in \{IAG, SAG, RAG, SPS\}$. Appendix F provides an overview of which pipelines were selected.

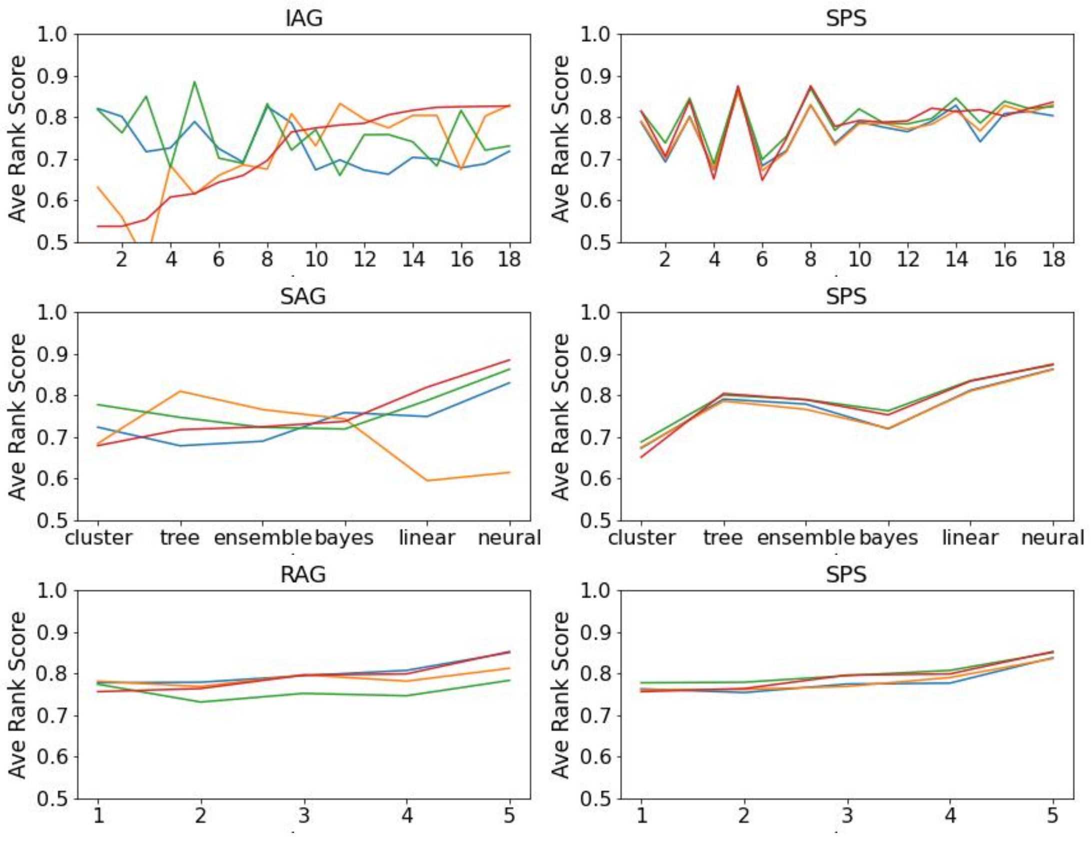


Figure 14: Performance of pipelines selected by the different pipeline selectors (IAG, SAG, RAG) for the groups specified by the individual pipeline selector for classification usecases. The performance of pipelines chosen by the SPS method on the groups of the corresponding grouping methods is displayed as a comparison. The difference in pipeline performance can be seen, whereby the grouping methods (left) produce more spread-out pipelines. The pipelines chosen by the statistical method (right) perform consistently over the different groups.

Figure 14 shows the performances of the different pipelines selected by the different pipeline selection methods for classification usecases. The figure shows the average performance scores of the pipelines selected by the corresponding selector on the groups of usecases described by that selector. As a comparison, the pipelines chosen by the SPS selector are plotted next to each grouping selector. The plots show that, as intended, the pipelines selected by the IAG and SAG selectors are more diverse, as individual pipelines perform better for individual usecases and are somewhat spread out. On the other hand, the SPS pipelines perform consistently well for each usecase group, with only minor inter-pipeline performance differences. As expected, the pipeline performances displayed by the pipelines selected by the RAG method resemble those chosen to be reliable. This result confirms that when grouping usecases to select top-performing pipelines to cater to individual strengths, the usecases should be grouped meaningfully rather than randomly. Doing so will negate grouping, and pipelines that perform well for all usecases are chosen.

Similar results were also found for the regression usecases (see Appendix G); however, looking more closely, a diverse split was only achieved for specific groups. A handful of pipelines selected by the

IAG selector, indicated by the numbers 15 to 17 (Gaussian Process Regressor, MLP Regressor and SGD Regressor), show similar rank performance across pipelines. The same behaviour is found for the usecases grouped by Bayesian algorithms in the SAG selector. This behaviour exhibits the same characteristics as reliable pipelines and might lead to a different meta-learner performance when training using the classification usecases and the regression usecases. A possible reason for this discrepancy may be attributed to the skewed nature of regression Scaled Rank distribution and will be addressed in the conclusion.

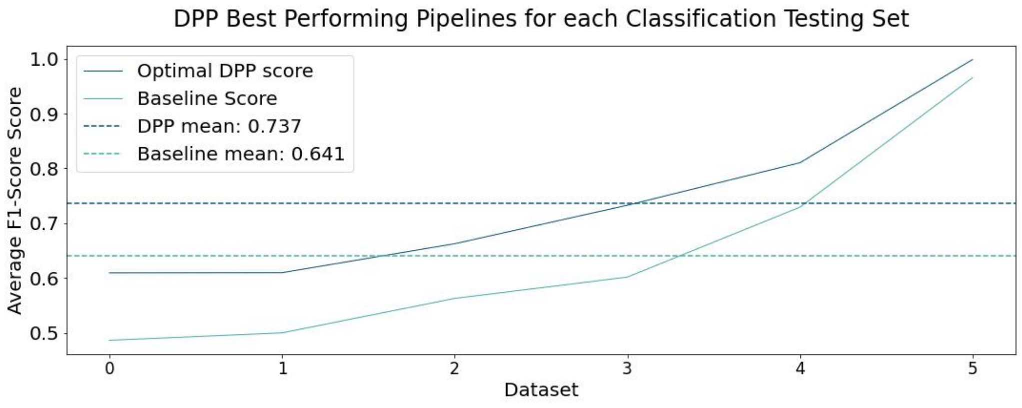
Besides finding pipelines ideal for only a subset of usecases and making the pipelines more distinguishable, as opposed to all usecases equally, the second idea motivating diverse pipeline selection was directed toward performance. Ideally, since a pipeline did not need to be universal and could be more specifically oriented towards solving the issues related to individual groups, the maximum performance attributed to diverse pipelines would be expected to outperform those attributed to reliable ones. This, however, was not accomplished by the grouping method pipelines. When the maximum values for each group score were compared, there was only a small difference between reliable and diverse pipelines for classification and regression usecases. The IAG pipelines achieved an average Scaled Rank score of 0.001 higher than the statistical pipelines, SAG pipelines achieving 0.01 higher and RAG 0.003 higher for classification usecases. For regression usecases, the average maximum Scaled Rank performance increased for the IAG, SAG and RAG selectors by 0.004, 0.001 and 0.017, respectively. Since the difference between the two methods is so small, the benefit of choosing diverse pipelines outperforming reliable ones in some instances was not found.

Combining Diverse and Reliable Meta-Targets The way the meta-targets are chosen using diverse or reliable meta-target selection methods is expected to have different benefits. Thus, it is natural to assume that using a combination of reliable and diverse meta-target together may be beneficial. There is, however, a danger of doing so. Since a reliable meta-target is likely to perform well for most usecases, compared to diverse meta-targets designed to maximise model accuracy of certain groups of usecases, a meta-model may begin only to recommend the reliable meta-targets over the diverse ones. The reason for this is that, by design, number of usecases that a diverse pipeline performs well for is much smaller than the number those a reliable pipeline performs well for. Thus, this project will not include a setting in which both meta-target selection methods are used simultaneously.

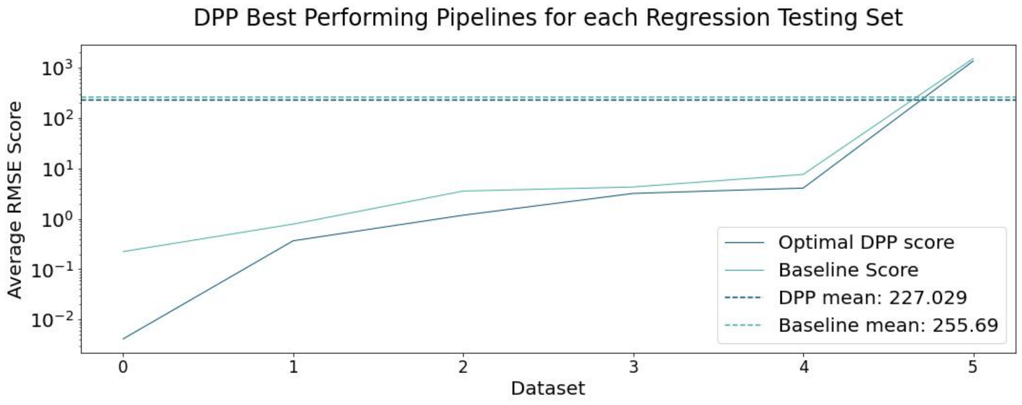
Number of Meta-Targets Only four different meta-targets were chosen for each meta-dataset, meaning the final model will only have four possible DPP pipeline recommendations depending on which meta-target selection method is used. This is a weakness of the system and something that can be improved on in future studies (more details as to possible future studies can be found in the Future Work section in Chapter 6.5). The reasoning behind choosing only to use four meta-targets was twofold. Firstly, it has become clear from previous studies that the prediction performance of meta-learning models aimed at predicting ideal DPP pipelines is a difficult task. Thus, having to predict more Scaled Rank scores from a larger pool of DPP pipelines would likely lower the prediction scores even further (Frye, 2023). Secondly, the diverse meta-target selection method was created with a low number of DPP pipelines in mind, making a selection based on top performers for multiple archetypes of DPP problems, as indicated by the groups. Due to the nature of meta-learning with DPP pipelines and the sheer number of possible meta-targets, a choice must be made regarding the number of meta-targets used. However, unless each possible DPP pipeline is included, a complete model is out of reach. This remains an issue with this method of meta-learning that future studies can improve upon.

3.2.3 Testing Sets

In order to draw valid conclusions about the performance of the final meta-learner, a meta-testset d_{meta}^{test} is required so that the model's performance can be evaluated on previously unseen data. To construct the new testing sets, additional information was collected by evaluating novel datasets and constructing further knowledge bases $\mathbf{K}_{test}^{t}, t = \{CL, RE\}$. Following the same methods outlined in the previous sections, the information contained within the testing knowledge bases was extracted to create a meta-testing set for each of the four pipeline selection methods. The meta-target pipelines were identical to those used in the corresponding meta-training sets.
In total, 12 new datasets were used in the testing set. These 12, split into six classification and six regression datasets, originated from eight publicly available datasets (see Appendix C). The same 18 ML algorithms were used in combination with the six testing datasets to create the testing usecases, leading to a total of 108 testing usecases for both classification and regression meta-testing sets. Like the training sets, the testing sets were shown to benefit from DPP pipeline transformations. The optimal DPP pipeline increased the average performance achieved by the models when no DPP pipeline was used (see Figure 15). More specifically, the average optimal increase that can be expected from using DPP pipelines was an improvement in F1-Score of 0.096 for classification datasets and an improved distance of 1.73 RMSE for regression datasets. Again, no tests of statistical significance could be carried out; however, like in Chapter 3.1.3, the results show a similar trend that DPP pipelines improved model performance.



(a) Classification Testsets



(b) Regression Testsets

Figure 15: Best performing pipeline F1-Score and baseline pipeline F1-Score averaged over each ML algorithm used on a single testset for each classification and regression testset.

3.3 Training

Referring back to the overview of the methodology of the project, shown in Figure 7, so far, the previous sections described how the information obtained from running many different ML algorithms was collected and transformed into a meta-dataset. The following section will explain how this data is exploited using a meta-model. In general, this meta-model is nothing more than a regular supervised learning ML algorithm that takes the meta-dataset as input and predicts the ranks of the pipelines in the candidate pool for each instance. Thus, the training of the meta-model follows the same procedure as any other ML algorithm optimisation approach. This includes an initial preprocessing of the inputs followed by the search for an ideal learning algorithm. Additionally, these learning algorithms were subjected to hyperparameter optimisation (see Figure 16 for an overview of the training process).

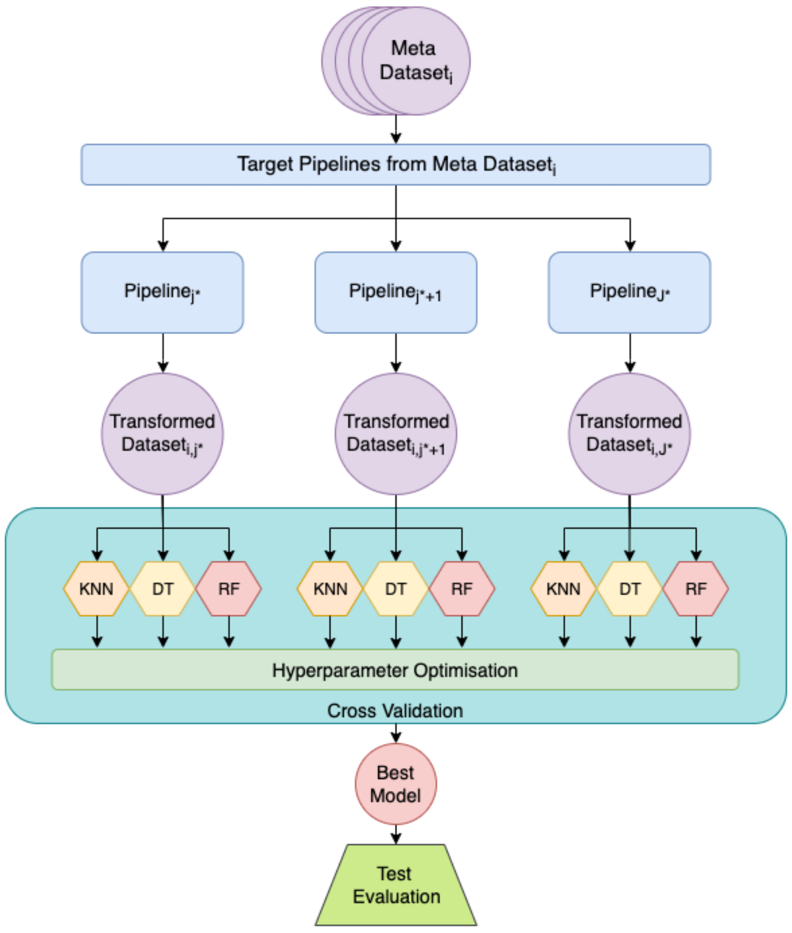

Meta Dataset$_i$
Target Pipelines from Meta Dataset$_i$
Pipeline$_{j^*}$
Pipeline$_{j^*+1}$
Pipeline$_{J^*}$
Transformed Dataset$_{i,j^*}$
Transformed Dataset$_{i,j^*+1}$
Transformed Dataset$_{i,J^*}$
KNN
DT
RF
KNN
DT
RF
KNN
DT
RF
Hyperparameter Optimisation
Cross Validation
Best Model
Test Evaluation

Figure 16: Overview of the meta-model training process involving preprocessing steps, hyperparameter optimisation and model selection (KNN, Decision Tree and Random Forest). The best-performing model will be saved and tested

The training procedure consisted of using cross-validation to achieve the best-performing model. Before training began, however, the meta-dataset was preprocessed, and an ideal ML algorithm was selected. The best model was determined by utilising a grid search over the search space of all training settings. This included the set of all meta-datasets, each produced by employing a different meta-target selection method $d_{meta,i} \in D_{meta}, i = 1,...,I = 4$, the set of all DPP pipelines that a meta-dataset would be transformed by $p_{i,j^*} \in P_i, j = 1,...,J^* = 4$, the set of ML algorithms used to train the multi-output regression model $\mu_a \in M, a = 1,...,A$ and the set of all combinations of model hyperparameters for a given ML algorithm $h_{\mu,k} \in H_\mu, k = 1,...,K_\mu$. In total, the number of cross-validations CV_{num} to be completed is $CV_{num} = I * J^* * \sum_{a=1}^{A}(K_{\mu_a}) = 2112$.

3.3.1 Preprocessing

The training process's goal is to attain the best model performance possible. From Chapter 2.2, it follows that the highest possible performance can only be achieved after the dataset undergoes DPP pipeline optimisation. To avoid adopting and running the collection algorithm from chapter 3.1.3, a more straightforward method was chosen to obtain well-performing DPP pipelines. Since the DPP pipelines selected to represent meta-targets by the meta-target selection methods were chosen based on high performance for either different algorithm groups or overall average performances, these meta-target pipelines were used to prepare the meta-datasets. Thus for each meta-dataset, $d_{meta,i}$, its respective target pipelines were chosen as preprocessing pipelines.
Once the pipelines were obtained, the meta-datasets could be transformed. However, due to some characteristics of the meta-datasets, not every method in the DPP pipeline was used to transform the datasets. For one, since the datasets had been created by hand, there were no missing values to be expected. This assumption led to imputation steps being redundant that were thus removed. The same can be said for encoders since the categorical meta-features had already been dealt with using the binary encoder.

3.3.2 Learning Algorithms

Since the focus of this project was to improve a meta-learning system using meta-target selection methods rather than better, more powerful models, the set learning algorithms that were chosen to learn the task of predicting the Scaled Rank score of DPP pipelines for different usecases were limited to commonly found ML algorithms used for many ML applications. Thus, the `Decision Tree Regressor`, the `KNeighbors Regressor` and the `Random Forest` models were chosen from the `scikit-learn` library.
Besides training each of the chosen models to determine which produces the best training result, different hyperparameter settings were applied to each model. A grid search was run over a selection of different hyperparameter settings since the number of hyperparameter settings was manageable. For an overview of the different hyperparameter settings, see Table 8.

3.3.3 Cross Validation

When training a supervised learning model, one must constantly be vigilant to avoid overfitting. Overfitting can occur through unfortunate/fortunate ways that a train test split can stack the data into producing better or worse results than are available in the ground truth. K-fold cross-validation can be employed as a countermeasure to this problem, which aims to avoid any outcomes dependent on the train-test split (Refaeilzadeh, Tang, Liu, et al., 2009). In K-fold cross-validation, the idea is to train and test the model for the different K folds in a dataset. Here, a fold refers to a subset of the data used as a validation set to obtain a validation score. The main idea with using K-Fold cross-validation is that each part of the data is the validation set, with the data outside of the fold being used to train the model. After the evaluation scores are obtained over each of the K folds, the scores are averaged, making the resulting evaluation score more reliable and less likely to be a result of overfitting.

ML Algorithm	Hyperparameters	Values
KNN	Weights	'uniform', 'distance'
	Algorithm	'auto', 'ball_tree', 'kd_tree', 'brute'
	Number of Neighbours	2,3,4,5,6,7,8
Decision Tree	Criterion	'squared_error', 'friedman_mse', 'poisson', 'absolute_error',
	Min Sample Required to Split	2,3,4,5
	Min Samples in Leaf	1,2,3,4,5
Random Forest	Criterion	'squared_error', 'poisson', 'absolute_error',
	Min Sample Required to Split	2,3,4,5
	Min Samples in Leaf	1,2,3,4,5

Table 8: Hyperparameter setting for each ML algorithm used as meta-models.

Usually, the folds are randomly selected; however, this is impossible for the current dataset. This is because certain instances or usecases in the meta-datasets depend highly on one another as a result of the dataset augmentation. When variations of datasets were created, making datasets with more missing values, more categorical features or both, their targets were still unchanged. Data leakage is inevitable if some of these instances were to be found, partially in a fold and partially in the training set. This pitfall was avoided by manually defining the folds. The methodology behind this manual definition was to insist that datasets with a common base (i.e. augmented from a shared dataset) are to always be together within a fold. A list of the folds used for the cross-validation can be found in Appendices H and I.

3.4 Model Assessment

As mentioned in the previous section, cross-validation scores were obtained by averaging a performance measure calculated using the predicted and actual values (validation values) over each fold. Using different performance measures makes it possible to choose the best-performing model depending on different model behaviours. To this end, selecting a performance measure to cater to specific needs is possible. For instance, depending on what is deemed ideal behaviour, a model that performs well while evaluating that behaviour can be chosen. With this in mind, three different performance measures were created to select models that perform well for three tasks: general Scaled Rank prediction accuracy, best pipeline recommendation and correct ordering of pipeline prediction ranks.

3.4.1 General Rank Accuracy

The accuracy of a regression model is evaluated via distance measures from the desired targets to the achieved targets. In the knowledge base (chapter 3.1.3), RMSE was used to obtain the distance score. This evaluation measure cannot be used in the meta-model since there is no single target, given that the model learns a multi-output regression task. Instead, the *average-RMSE* (ARMSE) was used.

$$ARMSE = \frac{\sum_{j^*=1}^{J^*} RMSE(y_{j^*}^{pred}, y_{j^*}^{val})}{J^*} \tag{6}$$

The ARMSE measure indicates the distance from the predicted Scaled Ranks to the actual Scaled Ranks in the training sets. A good measure suggests that the accuracy of the Scaled Ranks is high and that for each of the J^* candidate pipelines, the predicted Scaled Rank is, on average, close to reality. This measure is ideal for selecting models used to get a general idea of how well the candidate pipelines will perform on a new task. A high ARMSE, however, does not guarantee that ordering the pipeline ranks within the candidate pipeline pool is accurate. Different performance evaluations can pick models that perform well in those tasks.

3.4.2 Ideal Ordering

Another possible use of the meta-model would be to have it return a list of potential DPP pipelines in the order of predicted performance. As mentioned, having a model that accurately predicts the ranks of each of the pipelines on average may not guarantee a correct ordering of the pipelines, especially if the actual ranks of the pipeline candidates are similar. A method to help with this model application was designed to evaluate the model based on its ability to order the DPP pipeline ranks correctly.
Kendall's τ is a non-parametric measure of relationships between columns of ranked data. The Tau correlation coefficient returns a value between -1 and 1, with a 0 indicating no correlation between ranks, a 1 implying a perfect correlation and a -1 implying a perfect reverse correlation. To use this correlation coefficient, Kendall's τ was calculated over the actual and predicted rankings of the pipeline ranks of a given instance (usecase).

$$\tau_{u_i} = \frac{C_i - D_i}{C_i + D_i} \tag{7}$$

Where C is the number of *concordant pairs* or the number of times two elements are ranked in the correct order in both lists. In this case, the ranked lists refer to the predicted and actual ranks (element one ranked higher or lower than element two in both cases). Conversely, D is the number of *discordant pairs* or the number of times two elements are ranked incorrectly for both ranked lists.
To obtain a value concerning the model's ordering ability for the entire prediction, the $average - \tau$ ($a\tau$) was calculated by averaging the τ values over the prediction and validation scores for each usecases in the meta-dataset.

$$a\tau = \frac{\sum_{i=1}^{I} \tau_{u_i}}{I} \tag{8}$$

The value obtained from the $a\tau$ thus defines a model's ability to be able to predict the DPP pipeline ranks in the proper order correctly. This type of model performance may be an ideal choice when a subset of possible pipelines within the pipeline candidate pool is required.

3.4.3 Best Pipeline

Often, the most sought-after results when dealing with meta-recommendations are the ones that are likely to perform the best. Therefore, a further performance measure was implemented to select trained models that accurately predict the best-performing pipeline amongst the DPP pipeline in the candidate pipeline pool. First, the pipelines with the highest rank for the actual and predicted ranks for a given usecase of the meta-dataset were extracted. This resulted in two lists of DPP pipelines: the best-predicted pipelines and the best actual pipelines according to Scaled Rank scores. Then, to evaluate the model's ability to identify the best pipeline accurately, the F1-Score was calculated with respect to these two lists.

4 Experimental Setup

This project aimed to investigate the effect of using diverse meta-targets over reliable ones and whether doing so would positively affect meta-learning accuracy. Thus, a final meta-model was trained following the insights gained from the cross-validation results from the training phase and evaluating its performance based on the three evaluation criteria mentioned in Chapter 3.4. To reiterate, these consisted of measures testing the model's ability to predict accurate DPP pipeline ranks, to predict the order of the DPP pipeline performances correctly and to be able to determine which DPP pipelines performed the best correctly. Additionally, the effectiveness of the current meta-learner was assessed by comparing the performance of the selected pipelines to both a baseline performance and the performance achieved by a commonly used DPP pipeline recommendation system, TPOT. Two experiments were carried out, each addressing one of the previous questions.

4.1 Experiment 1: Pipeline Selection Evaluation

The first experiment determined the effectiveness of diverse meta-target selection methods compared to reliable selection methods. This was achieved by comparing the performance of the best diverse pipeline selector, either the *Individual Algorithm Grouper* (IAG), the *Similar Algorithm Grouper* (SAG) or the *Random Algorithm Grouper* (RAG), to the performance of the traditional *Statistical Pipeline Selector* (SPS) which was shown to choose reliable pipelines. The best pipeline selector was chosen by analysing the results obtained by training the meta-model using cross-validation. Since three different evaluation methods are considered, the ideal settings, including the ideal diverse pipeline selector, were chosen for each evaluation metric. Once the best model settings and best pipeline selectors were identified, a subsequent model was trained using the full meta training set d_{meta} and tested on the meta testing set d_{meta}^{test}. Finally, a comparison was made between the performance of the final model that predicts the ranks of reliable pipelines compared to the final model trained to predict diverse pipeline ranks.

4.2 Experiment 2: Recommended DPP Pipeline Performances

The second experiment carried out in this project evaluated the current model's usefulness and its ability to recommend DPP pipelines that enhance the performance of ML models. This was accomplished by obtaining model performances after transforming testing datasets using recommended DPP pipelines compared to the baseline (not using any DPP pipeline). Additionally, the performance achieved by a commonly used alternative AutoML system that can be used to recommend DPP pipelines was collected to test the viability of the current model through comparison.
The AutoML system that was used to compare the pipelines recommended by the model developed in this project was the TPOT model developed by Olson et al. (2016). Although more modern approaches to DPP recommendation exist, a handful of issues lead to using TPOT exclusively as a benchmark comparison, explained in the Limitations section in Chapter 6.4. As mentioned in Chapter 2.3.1, TPOT can be used to optimise the entire ML pipeline, including hyperparameter optimisation. However, since the current model was only trained to suggest DPP pipelines and does not take hyperparameters of the different ML algorithms into account, this feature was removed from the optimisation process. Besides hyperparameters, however, TPOT performed model selection and DPP pipeline optimisation. The learning algorithms considered are the same ones used in the usecases (see Table 3). However, each possible DPP method, including DPP method hyperparameters available for TPOT, was included in the search space (see Appendix L for an overview of the DPP methods).

5 Results

In the following section, the results of the project will be evaluated. Since not only the hyperparameters of the final meta-model were obtained from the cross-validation but also the optimal meta-target selection methods for each evaluation measure, the analysis of the cross-validation results will be analysed prior to the three experiments in Chapter 5.1. Naturally, the cross-validation results are obtained using only the training data and the averaged validation values across the different folds, as mentioned in Chapter 3.3.3. On the other hand, each of the experimental results was obtained by training the final-meta models using the entire training data, and testing using the testing sets described in Chapter 3.2.3. The outcomes of Experiment 1 and Experiment 2 will be presented in Chapters 5.2 and 5.3 respectively.

5.1 Cross Validation Results

The results obtained from the cross-validation will be analysed to choose the best pipeline selectors for use in the training of the final models. The cross-validation results were obtained through training and validating the different meta-datasets described by the pipeline selection method used to obtain target variables. Different multi-output regression models were trained using cross-validation for each evaluation and pipeline selection method. After training, models that produced the best cross-validation results were saved, along with the different DPP and hyperparameter that led to the highest performance. This collection of models resulted in the best models for each of the different ML algorithms, which could then be compared. This process was repeated for the different measurements and pipeline selection methods. Due to the large amount of time required to obtain the cross-validation scores for each meta-learner configuration, this procedure was only performed once. This means that no significance values could be calculated between the differences in meta-learner performance since only one result per meta-learning strategy was collected. Nonetheless, the outcomes of the cross-validation suggest which models and model hyperparameters should be used for the subsequent experiments.
The results of the best-performing models can be found in Table 9. At first glance, the results highlight the difficulty of learning an accurate representation of how different DPP pipelines will lead to different cross-validation results depending on dataset meta-features. This finding is indicated by an optimal ARMSE value of 0.179 for classification usecases and 0.197 for regression usecases achieved by the SPS selector. Since the ARMSE values can be interpreted with respect to the targets themselves (see Chapter 2.1.3), the accuracy of the meta-learner has a minimum error of approximately 18%. The ARMSE cross-validation results suggest that this error's size depends on the method of pipeline selection. Where the spread of the possible Scaled Rank scores is small, so too is the general error of the predictions. Besides the RAG method, which achieved the second-best score, confirming that the RAG selector will behave similarly to the SPS method, the remaining selection methods performed worse (see Appendix J for a complete overview of the cross-validation results). The better of the two was the SAG method for classification usecases and the IAG method for regression usecase. These results support the idea that different meta-target selection methods affect the outcome of meta-learning.
Unlike the reliable pipeline selectors delivering better results for the ARMSE distance measure, the diverse pipelines were shown to be more successful when the order of the predicted pipeline ranks was assessed. The results for both the F1-Score and the $a\tau$ measure suggest that having diverse pipelines can enhance the ability of a meta-learner to predict the performance of distinct targets in relation to others. For classification pipelines, the pipeline chosen by the SAG selector leads to the

Measurement	ML Task	Best Selector	Best Algorithm	Score
ARMSE	Classification	SPS	Random Forest	0.17994
	Regression	SPS	Random Forest	0.19732
$a\tau$	Classification	SAG	Decision Tree	0.15166
	Regression	IAG	Decision Tree	0.29487
F1-Score	Classification	SAG	Decision Tree	0.31945
	Regression	IAG	Decision Tree	0.3493

Table 9: Best cross-validation results for each performance measurement, pipeline selection method and ML algorithm. See Appendix J for a complete overview of all cross-validation outcomes.

best meta-model $a\tau$ and F1-Score values; for regression pipelines, the same was achieved by the IAG selector. The difference between the two pipeline selectors for regression and classification usecases, specifically the lower scores achieved by the SAG selector for regression usecases, may be attributed to the spread of the selected pipelines in the SAG selector (Figure 17).

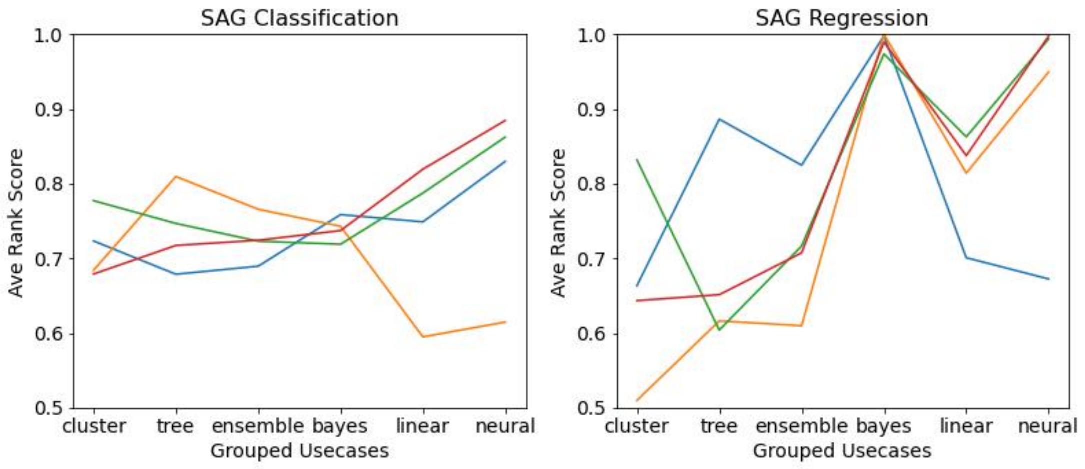


Figure 17: Difference between the selected pipelines for the SAG selector in classification and regression usecases. The plots show the differences in the spread of the selected pipelines. Specifically in the Bayes and Neural Network groups, the spread of the pipeline results is less diverse in the regression usecases compared to the classification usecases, which may have led to the lower $a\tau$ and F1-Score cross-validation results.

To summarise, the cross-validation results show that reliable pipeline selection outperforms diverse pipeline selection in a meta-model's capability to predict the rankings of DPP pipelines as the predicted values are nearer to the actual ranks of the DPP pipelines. This is likely due to the range of the possible ranking values having a smaller spread. In contrast, the diverse pipeline selection methods are better suited for tasks that predict which pipelines are better or worse than other pipelines given a certain usecase. Since the question of whether diverse or reliable pipelines will lead to greater model performances remains, the best reliable and diverse selector will be chosen for the subsequent experiment. Thus, for classification usecases, the SAG selector will be used. For regression usecases, the IAG selector will be the diverse selector. Finally, the SPS selection method will be the reliable selector for classification and regression usecases. Concerning the choice of ML algorithm for the final meta-model, the cross-validation results show that the Random Forest is best suited for DPP rank prediction and Decision Trees are best suited for predicting DPP pipeline orders. An overview of the remaining meta-model settings, including hyperparameter settings and DPP pipeline transformations, can be found in Appendix K.

5.2 Reliable vs Diverse Selection Method

The first experiment's results were collected to answer whether reliable or diverse meta-targets will lead to better meta-learner performances. For classification and regression usecases, meta-models were trained with reliable or diverse targets and using the optimal model configurations obtained from cross-validation for a total of 30 times. These configurations included the ideal DPP pipeline, ML algorithm and hyperparameter setting for each performance measurement. The ideal model configurations can be found in Appendix K. In addition, the resulting scores can be found in Table 10.

ML Task	Measure	Reliable		Diverse		Difference
		Mean	Std	Mean	Std	
Classification	ARMSE	0.2005	0.004	0.1974	0.0028	**-0.0031****
	$a\tau$	0.0606	0.0409	0.1481	0.0312	**0.0875****
	F1-Score	0.2471	0.0606	0.3282	0.0589	**0.0811****
Regression	ARMSE	0.267	0.007	0.2879	0.0029	0.0209
	$a\tau$	0.1225	0.0699	0.1037	0.0336	-0.0188
	F1-Score	0.3218	0.0617	0.2304	0.0529	-0.0914

Table 10: Mean and standard deviation of test results for meta-models that predict reliable or diverse meta-targets. The difference between the two scores is also included with an improvement from reliable to diverse pipelines, indicated by a positive value for $a\tau$ and F1-Score measures and a negative value for ARMSE. For clarity, improvement is marked in bold. A significance value of $\alpha^{*} = 0.05$ and $\alpha^{**} = 0.01$ were used; however, after using a Bonferroni correction to account for running multiple significance tests, these values were updated to $\alpha^{*} = 0.0083$ and $\alpha^{**} = 0.0016$.

As mentioned in Chapter 3.2.2, it was hypothesised that selecting diverse pipelines would lead to better meta-model results than reliable pipelines, as having larger differences in pipeline performance would allow pipelines to be better distinguishable. The results show that significant improvements were achieved for classification usecases. Most notably, the $a\tau$ and the F1-Score improved by 0.087 and 0.081, respectively. Interestingly, the ARMSE value also improved, though the difference between the ARMSE score attained by the diverse pipelines over the reliable pipelines is slight.
These results do not carry over when looking at the performance achieved by the meta-model using pipelines for regression usecases. In contrast to the classification results that showed no improvement could be observed when using diverse meta-targets over reliable ones. In fact, each evaluation metric showed a decrease in performance. Since the results obtained by using regression pipelines contradict those obtained by using classification pipelines hypothesis of diverse pipelines outperforming reliable ones when training meta-learning systems can only be partially confirmed. A possible reason behind the discrepancy between the two results may be attributed to the problems noticed in Chapter 3.2.2 with respect to regression pipelines selector only selecting pipelines that are diverse for certain groups of usecases for both the SAG and IAG selectors. Further investigations on the issue will be carried out in the discussion.

5.3 Recommended DPP Pipeline Performance

The final experiment was aimed at testing the practicality of the current meta-learner by testing the DPP pipelines that were the most highly recommended from the set chosen by either meta-target selection method. The Scaled Rank scores of these recommended pipelines chosen by the best diverse and reliable selectors are first compared to the Scaled Rank score achieved when not using any DPP pipeline to transform the datasets. Figure 18 shows all DPP pipelines' rank performances after transforming each testing set. Additionally, the baseline performance and the average Scaled Rank score of the pipelines that performed best are indicated. The results show that, should the maximal pipeline be chosen to transform the datasets, the performance is much higher than when no DPP transformations are used.

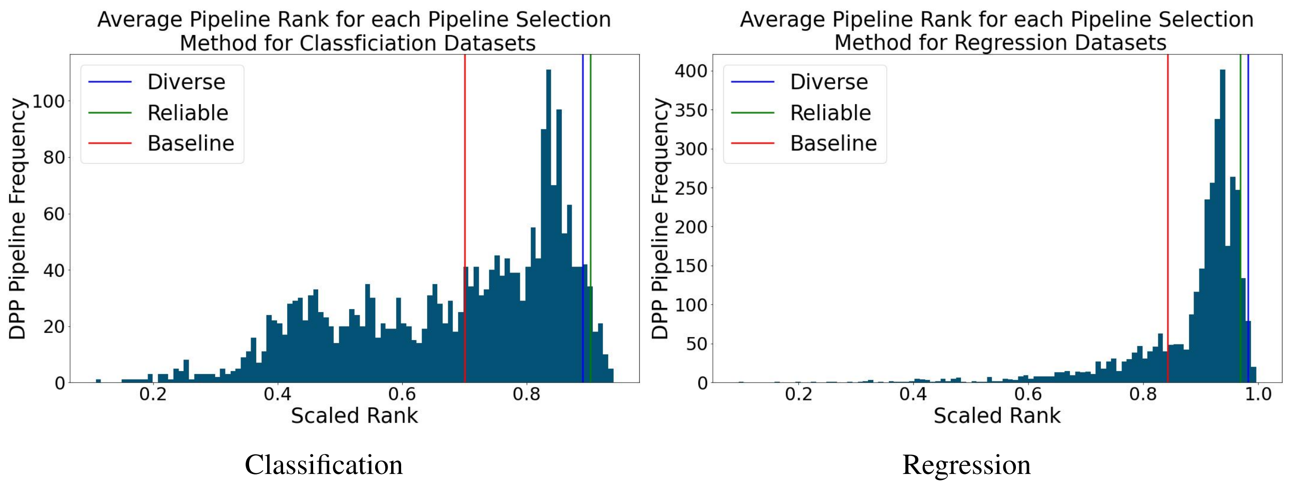


Figure 18: The average Scaled Rank score achieved by the best-performing pipelines chosen by the reliable meta-target selector and diverse meta-target selector compared to the average performance when no DPP pipelines are used to transform the classification and regression test-usecases. It can be seen that a considerable benefit can be obtained by using the DPP pipelines recommended by the current model.

Besides comparing the potential performance that can be achieved by the pipelines selected by the pipeline selectors to the baseline scores, the following test aims to compare the performance of these pipelines to those recommended by a commonly used external DPP recommendation system. First, the popular DPP optimisation tool, TPOT, was used to find DPP pipelines for each dataset using the classification and regression usecases in the testing sets. Then, these pipelines were compared to ones obtained from the current meta-learning system. The pipelines with the highest predicted Scaled Rank score and the best possible pipeline for each usecase from either the set of diverse or reliable pipelines were used for this comparison. The median scores over the different Ml algorithms for each dataset were gathered for the baseline, current model and the TPOT results. As before, the F1-Score was used to evaluate the performances of the classification dataset sets. However, to improve interpretability when looking at the results for regression datasets, NRMSE values were used over the regular RMSE. The results of this experiment can be found in Table 11.

$$NRMSE = \frac{RMSE}{maxValue(dataset_{targets}) - minValue(dataset_{targets})} \tag{9}$$

It can be seen that not only do the DPP pipelines that achieved the optimal score per usecase selected by both reliable and diverse outperform TPOT, but also those that were predicted by the meta-learner.

This outcome shows that this method of DPP pipeline recommendation can be used reliably in a real-world setting, despite the low F1-Scores achieved by the meta-models (see Table 10). When looking at the difference in performance between the reliable and diverse pipelines, the results suggest that reliable pipelines lead to slightly better dataset transformations than those chosen by diverse meta-target selection methods.
It is important to note that none of the differences measured in this experiment was statistically significant. For example, even the difference between using no DPP pipeline and the optimal, reliable pipelines for classification and regression usecases achieved a p-value of 0.162 and 0.151, respectively. This lack of statistical significance leads back to only using six testing datasets. Therefore, a lack of statistical power was present and decreased the likelihood of obtaining significant results. Thus, no claims can be made about significant differences in observed performance between results in this experiment. However, the observed scores suggest what could be expected from the different models if more datasets were used.

Classification Dataset	Baseline	ML-DP	ML-RP	ML-DO	ML-RO	TPOT
Pd Catalyst	0.5111	0.6489	0.6771	0.6595	0.6711	0.5869
Sea Temp CL	0.7883	0.8037	0.8442	0.8306	0.8482	0.7867
Segment	0.9864	0.9934	0.9903	0.9967	0.9935	0.9908
Pd Decorated Catalyst	0.5313	0.6556	0.6815	0.6597	0.6841	0.5591
Activation Humidity	0.5891	0.6674	0.6693	0.6882	0.6518	0.6596
Pt Catalyst	0.6255	0.7276	0.7607	0.772	0.7863	0.6311
Average	**0.6720**	**0.7494**	**0.7705**	**0.7678**	**0.7725**	**0.7024**
Std	0.1828	0.1331	0.1271	0.1312	01322	0.1619

Regression Dataset	Baseline	ML-DP	ML-RP	ML-DO	ML-RO	TPOT
Cement Strength	0.1074	0.0736	0.0684	0.0711	0.0706	0.1003
WaterWasteKws	0.0295	0.031	0.0346	0.0291	0.0291	0.0374
Sea Temp RE	0.0922	0.0832	0.0815	0.0691	0.0691	0.0821
NationalCapability	0.0522	0.0263	0.0271	0.0263	0.0253	0.0468
Used Cars	0.0822	0.0823	0.081	0.0748	0.0748	0.0794
WaterWasteGallons RE	111.5009	0.1172	0.0937	0.0912	0.0912	0.1138
Average	**18.644**	**0.067**	**0.062**	**0.0603**	**0.06**	**0.0766**
Std	45.4904	0.0342	0.0261	0.0264	0.0267	0.0297

Table 11: Overview of DPP pipeline results on classification and regression testing sets (F1-Score for classification datasets and NRMSE for regression datasets). The results for the baseline scores (no DPP pipelines used), Meta-Learner-Diverse Predictions (ML-DP), Meta-Learner-Reliable Predictions (ML-RP), Meta-Learner-Diverse Optimal (ML-DO), Meta-Learner-Reliable Optimal (ML-RO) and TPOT Pipelines are shown.

6 Conclusion

In this project, a meta-learning model was designed to predict the performance of several DPP pipelines for incoming datasets and ML algorithm pairings (usecases). The proposed model was created by gathering information regarding the performance of many DPP pipelines on various datasets and ML algorithms. Subsequently meta-datasets were created to train a multi-output regression meta-learning model. The goal was to investigate whether different meta-target selection methods could enhance the prediction performance of the meta-model. Thus, two research questions were set that would be answered through different experiments.

6.1 Meta-Target Selection

The first question investigated in this project aimed to explore alternative meta-target or candidate DPP pipeline selection techniques. Whereas the standard method for selecting DPP pipeline meta-targets was based on selecting reliable DPP pipelines, DPP pipelines with an overall high average performance (Laadan et al., 2019; Zagatti, 2021), this project introduced an alternative pipeline meta-target selection method. The method in question aimed to select diverse DPP pipeline meta-targets. Instead of selecting targets that perform consistently well across the individual usecases, the alternative approach involved choosing a number of DPP pipelines that would each perform exceptionally for different groups of usecases. The motivation for diverse pipeline selection was two-fold. Firstly, locate DPP pipelines that lead to maximum performance for specific types of usecases that would otherwise not be considered due to low average performance across all usecases. Secondly, find meta-targets that are more distinguishable to increase meta-learning prediction accuracy.

Three evaluation measures were used to test the performance of the meta-learner to investigate the effects of different pipeline selection methods. The first of which, the ARMSE value, inspects the distance from the predicted ranks to the actual ranks in the testset and is seen as the general accuracy of the model. The remaining methods focus on the ordering with the $a\tau$ value inspecting the model's accuracy in predicting the ranks of the DPP pipeline meta-targets in the correct order and the F1-Score inspecting the model's ability to determine the best-performing pipeline. Since reliable selection methods were expected to select pipelines with only small inter-pipeline fluctuations between usecases, limiting the amount of spread observed in the DPP pipeline ranks and thus in the pipeline rank predictions. Thus, reliable pipelines were expected to perform better than diverse pipelines. In contrast, since the remaining two measures benefit from better distinguishing between pipelines, it was expected that using diverse pipelines would improve the meta-learning results for $a\tau$ and F1-Score.

6.1.1 Classification Usecases

The results of Experiment 1, documented in Chapter 5.2, show that using diverse pipelines for multi-output meta-targets affects the meta-learning prediction performance. It was found that the performance of the meta-learner improved for each of the evaluation measures for classification usecases. For these usecases, the trained meta-learner significantly improved its performance when selecting the best-performing pipeline and predicting pipeline ranks in the correct order (see Figure 10). Additionally, a slight improvement was observed in general rank prediction accuracy. This outcome suggests that using diverse pipelines instead of the standard methodology of selecting reliable pipelines may be worth pursuing and investigating in the future.

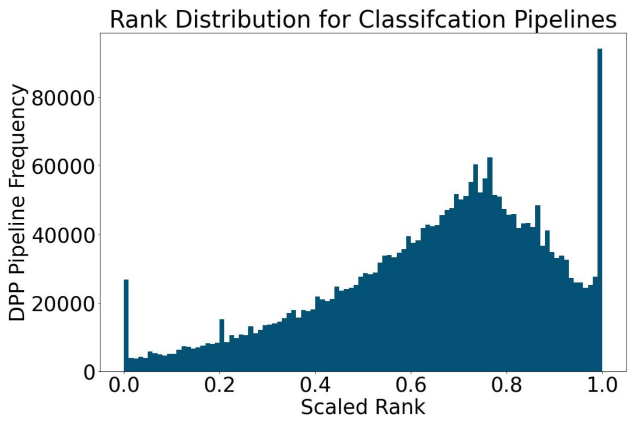
Some possible explanations can be made about why diverse pipelines performed better than reliable pipelines for classification usecases. First, it was argued that they cause pipelines to be more distinguishable, as reliable pipeline ranks fluctuate around smaller ranges. However, for many well-performing ML algorithms, the target values' scale and range is not an essential factor and can be trained successfully, given enough training examples to solve the problem efficiently. If only a few training examples are available or the learning task is difficult, on the other hand, the accuracy of a meta-model may be low. In such cases, making the targets that are to be predicted more distinct can improve its performance. The final meta-model results show that the general error, as conveyed by the ARMSE scores, is relatively high. The model's accuracy produced rank predictions with an average error of approximately 20%. Thus the benefit of predicting pipeline ranks that are further apart from each other increases the likelihood of the predictions being representative of the ground truth as the error tolerance is larger. With this in mind, it may be worth further investigating in what situations diverse meta-target selection can improve meta-learning and if it can be employed to boost the learning performance when training data is difficult to obtain or when the learning tasks are challenging to learn.

6.1.2 Regression Usecases

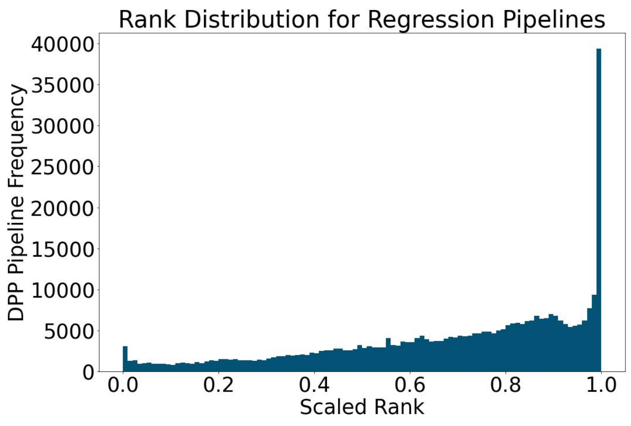
The findings observed when looking at the results for the classification usecases were not reflected when predicting the ranks of the regression usecases. Here, the meta-learner performed slightly worse for all three evaluation metrics when using diverse meta-target selection methods compared to diverse ones. As mentioned previously, a possible reason for this lower performance might be due to the pipelines that the diverse pipeline selector selected. Figure 17 highlights the disparity between the selected diverse pipelines for classification and regression usecases. It can be seen that where the pipelines for classification usecases are more clearly distinguishable for each group, those selected based on regression usecases have low inter-pipeline performance differences.

An explanation for why the diverse meta-target selector does not find pipelines of the same quality as those selected based on classification usecases can be found when looking at the distribution of the Scaled Rank scores achieved by every pipeline on each usecase. When comparing the Scaled Rank distribution of classification pipelines (Figure 19a) to the Scaled Rank distribution of regression pipelines (Figure 19b), it can be seen that the regression ranks are skewed towards the maximum Scaled Rank score of 1.0. Though many pipelines are also found to reach the top of the distribution for the classification pipelines, the Scaled Rank scores are more evenly distributed than with regression pipelines. Since the diverse pipeline selector selects pipelines based on maximum Scaled Rank values, it becomes likely that more homogeneous pipelines will be selected for regression usecases.

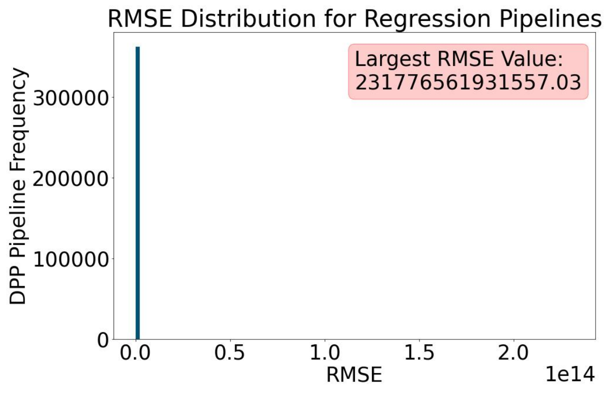
To investigate why regression pipelines are found at higher Scaled Rank values than classification pipelines, the distribution of RMSE values, from which the Scaled Ranks are established, is shown in Figure 19c. Due to the nature of RMSE values being highly dependent on the scale of the targets in the dataset, we can see a considerably large spread of RMSE scores. The vast majority lies close to 0.0; however, some RMSE scores are much larger, with the maximum RMSE value reaching an extreme value of $2.32 * 10^{14}$. As the Scaled Rank scores maintain the original shape of the distribution of the RMSE scores (see Figure 11), it becomes clear why so many Scaled Rank scores are close to the maximum. The obvious solution to this problem would be a method of outlier removal. Employing such a strategy, however, raises further problems. After further investigation, it was determined that many of these extreme values are not outliers, as the third standard deviation for RMSE Scaled Rank scores is $1.43 * 10^{12}$. Removing all values greater than this would not change the distribution meaningfully. Even if the range of RMSE values is restricted between 0 and 1, the distribution of scores is still highly skewed while losing over 44% of the total pipelines within the knowledge base (see Figure 19d). Thus, this issue seems to be fundamental to meta-learning for regression usecases with the RMSE as the performance metric.



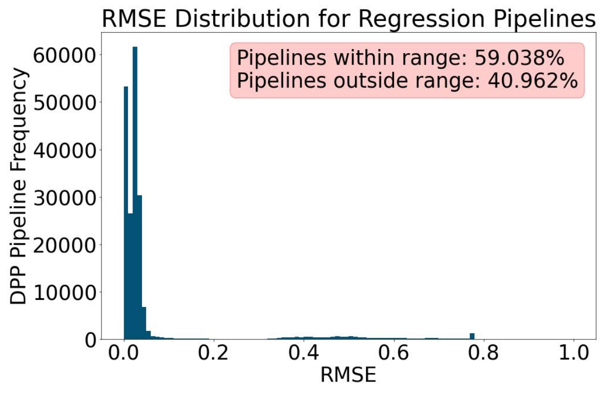
(a) Scaled Rank distribution for classification pipelines



(b) Scaled Rank distribution for regression pipelines



(c) RMSE distribution for all regression pipelines



(d) RMSE distribution between 0.0 and 1.0

Figure 19: Graphs aiding the investigation of the difference in meta-target selection results between classification and regression training usecases. Figures a and b show the difference in the shape of the Scaled Rank distribution, showing that the Scaled Rank scores for regression usecases are heavily skewed to the maximum value. An explanation for the shape of the Scaled Rank distribution can be found when looking at the distribution of the RMSE scores for regression usecases. This RMSE distribution is shown in Figure c. It shows that, due to differences in target variable scales between datasets, most values lie close to 0.0, with a large spread reaching much larger values. In Figure d, the distribution of the regression usecases is shown in a range between 0.0 and 1.0. Here, the distribution is still heavily skewed; however, over 44% of the data points have been removed, ruling out outlier removal as a tool to solve the issue easily.

A solution to this problem is to use the NRMSE instead of the RMSE value within the knowledge bases. The NRMSE was not used over the RMSE value originally because it was expected that the Scaled Rank conversion would solve the issues related to scaling sufficiently. However, these issues permeate more severely than expected. Using this metric over the RMSE value would allow the Scaled Rank values to be more evenly distributed around less extreme bounds, allowing for much easier comparison between datasets that would benefit pipeline selection methods. An alternative approach to solving the scaling issue between datasets would include a mandatory scaling of the regression target value before the model evaluation collection. This method is more drastic than normalising the response value since this changes the actual target values of datasets and may have unforeseen consequences when wanting DPP pipeline recommendations for regression datasets with unscaled target variables. Nonetheless, the net gain may outweigh the downsides of using RMSE values in the knowledge base, as observed in this project.

6.2 Meta-Learner Viability

Experiment 2 was directed at testing the usefulness of the model created in this project and answering whether using this meta-leaner would be a viable alternative to currently available DPP pipeline optimisation solutions. This was done by comparing the performance of both the optimal pipelines, as well as those that were predicted to have the highest Scaled Rank for given usecases (dataset and ML algorithm pairings) to the baseline results (not using any DPP pipelines), as well as the TPOT model optimisation tool limited to optimising DPP pipelines. The results, found in Table 11, show that the potential benefit that the current meta-learning system can achieve is substantial. Both the optimal pipelines within the reliable and diverse pipeline pool and those predicted show greater accuracy improvements than the baseline and those achieved by the DPP pipeline optimisation ability of TPOT. This suggests that meta-Learning systems can compete with modern DPP pipeline optimisation tools when aiming to complete a DPP pipeline optimisation task. These results were found for both the classification and regression testing sets.

6.2.1 Diverse vs Reliable Recommendations

In Chapter 3.2.2, it was theorised that when selecting meta-tarts using a diverse meta-target selection procedure, the eventual pipelines in the resulting set will be more specialised toward different types of datasets than pipelines, selection using a reliable approach. The reason for this expectation was that a set of diverse meta-targets contain pipelines that only aim to score high for some groups of usecases while disregarding the scores achieved for others. This creates a collection of pipelines, each with an ideal type of usecase that can be processed exceedingly well. For a reliable pipeline to be selected, on the other hand, it must perform well for all usecases leading to a set of similarly good tools that can be used interchangeably for incoming usecases.

When comparing the resulting evaluation scores observed after transforming the testing usecases using the best fitting pipeline within the set of selected pipelines, we see that the reliable pipeline performs better than diverse ones (see Table 11). Disregarding the performances for regression usecases due to the scaling issues that may have led to opposing results in Experiment 1 (see Chapter 6.1.2), the current results suggest that, despite diverse meta-target selection leading to an increase in meta-learner accuracy, this technique is not guaranteed to improve DPP pipeline recommendations. A potential reason for the lower performances could be due to the current model only using four meta-targets. This restriction may have been too severe, forcing the pipelines within the set of diverse meta-targets to perform well for a number of usecases that was too large. In future work, it would be beneficial to investigate the effect of increasing the number of meta-targets when using diverse meta-target selection to see if the recommended pipelines can outperform those chosen by reliable meta-target selection methods.

6.3 Summary of Main Contributions

This project aimed to develop a DPP pipeline performance prediction meta-learner that can be used as a DPP pipeline recommendation tool. More importantly, this project investigated whether alternative meta-target selection methods could increase meta-model prediction performance. These methods focused on highlighting the difference in performance when the selected meta-targets were reliable or diverse. If they were reliable, the ranks attributed to the selected DPP pipelines were consistently high on average across each usecase. On the other hand, when the meta-target selection method favoured diverse DPP pipelines, a set of meta-targets were chosen such that each target becomes a specialist for only particular groups of usecases, allowing for greater differences between the meta-targets.

The overall results of the project indicate that a diverse meta-target selected can improve the performance of meta-learning systems by making the learning task easier, especially when the learning task is difficult and limited training data is available. Furthermore, the results showcase that pipelines predicted by the current meta-learner perform well compared to modern DPP pipeline optimisation tools and when not using DPP pipelines. However, when comparing the recommended pipelines' usefulness from diverse meta-target selection methods to those originating from reliable meta-target selection methods, the results show that reliable pipelines lead to more favourable transformations than diverse pipelines. Thus, although diverse meta-target selection does increase meta-target prediction accuracy, the pipelines chosen by the current version of the diverse method do not lead to better DPP pipeline recommendations.

6.4 Limitations

Several problems occurred during this project that would have to be addressed in future studies. The first significant issue that, if solved, could lead to more prominent meta-model performance was the small training set size. Since the knowledge base was taken from Frye (2023), which focuses mainly on creating DPP pipeline recommendation systems for production datasets, the pool of available datasets was naturally smaller. This issue was addressed by augmenting the datasets by adding missing values or categorical columns or employing both augmentation strategies. However, doing so led to many similar meta-features and performance scores and not adding much variety to the training data. Having so few fundamentally different datasets may have been the cause for the high difficulty level of the learning task, indicated by the relatively poor evaluation metric scores. Future studies would benefit greatly from a more extensive and less exclusive dataset collection.

Another area for improvement that resulted from the collection of the information contained in the knowledge base was the fact that several erroneous model evaluations were encountered when producing the evaluation results for the extensive list of DPP pipelines. In certain circumstances, an error was made when calculating either the RMSE scores or F1-Scores for a particular DPP pipeline, resulting in missing values for these specific cases. This led to subsequent problems during the creation of the meta-datasets. Because of the errors made during this collection period, a pipeline selected by one of the meta-target selectors contained missing values for specific usecases. The usecases had to be removed in such a situation since no target Scaled Rank score could be learned. Given that the Scaled Rank scores of several DPP pipelines were required, the chance of encountering target variables with missing values grows as the number of meta-targets increases. In the future, developing a more robust knowledge base collection method would be beneficial to produce fewer missing values.

A new method for knowledge base evaluation collection should also account for the loss of information that is traced back to using only the default learning algorithm hyperparameters during the collection of model scores. Since model performances and general behaviour depend severely on the choice of hyperparameter settings, the performances of the different DPP pipelines will also be vastly different. This means that once model hyperparameters are changed, the recommended DPP pipeline a user has received using the current model will not be directly applicable. One way that a future project could solve this issue would be to include each different hyperparameter configuration in the model evaluation collection process. The severe increase in necessary computation time could be balanced by limiting the number of different learning algorithms used.

A further issue that led to inconsistent results is the nature of regression scores within the regression knowledge base and the Scaled Ranks produced. As previously discussed, the distribution of the Scaled Rank scores for all DPP pipelines was highly skewed towards the maximum, resulting from using RMSE values on unscaled target values within the regression datasets contained in the knowledge base. Since the RMSE values depend on the regression targets' scale, the evaluation results collection was initially on vastly differing scales. This issue led to the disparities between diverse pipeline selection for regression usecases compared to classification usecases. Future studies on meta-learning systems using knowledge bases should consider using NRMSE response variables or including a target value scaling to the set of mandatory transformations during the collection phase to overcome these issues.

An improvement could also be made during the training period of the meta-models. For example, when determining ideal ML algorithms, DPP pipeline transformations and hyperparameter tuning, this project employed a grid search for each training model. To save time, this grid search was also kept relatively small. However, to achieve better-performing model settings, either a more extensive search space would be required, or better still, and more in line with the source material, automatic optimisation tools, such as Bayesian Optimisation, could be deployed (Klein, Falkner, Bartels, Hennig, & Hutter, 2016).

Finally, the benchmark that the performance of the recommended DPP pipelines selected by the current model was only compared to those selected by TPOT. This is naturally only a partially satisfactory comparison since newer, more popular DPP pipeline selection methods have become available. The two most accessible models that could have been used for comparison are `AutoSklearn` and `AutoClean`. Unfortunately, some problems hindered the employment of said models. For one, as the current meta-learner was only used to obtain DPP pipeline recommendations, the performance obtained used to evaluate the effectiveness of the chosen DPP pipelines was done without optimising hyperparameter settings. This limitation does not align with the options available when using `AutoSklearn` since the search space always includes various ML algorithms and hyperparameter settings. Thus, the comparison would not have been accurate. Regarding `AutoClean`, the current implementation was unstable at the time of writing, and a working version could not be run. Thus, this paper was limited to using TPOT as a comparison.

6.5 Future Work

Besides improving the issues mentioned in the previous section, several findings in this project warrant further investigations. For one, it was found that a diverse meta-target selection method can increase the performance of meta-learning models. However, it is unclear under which conditions diverse meta-targets are more beneficial. For this reason, future research focusing on the difficulty of the meta-learning task would be required. By varying the task difficulty, it is possible to determine the conditions that warrant diverse meta-target selection methods necessary over reliable meta-target selection. Additionally, the method for selecting diverse meta-targets could be improved in future research. It could be possible to increase the robustness of the selected pipelines so that, even without scaling regression target variables, sufficient diverse pipeline selectors will perform as well for

regression as they do for classification.
An additional point that could be looked into in future studies would be the effect of the number of meta-targets on the performance and overall DPP pipeline recommendation of a multi-output regression meta-model. It would be expected that the resulting evaluation metrics would worsen with an increased number of meta-targets. However, the resulting DPP pipeline recommendations would likely improve given that having a large sample of possible recommended DPP pipelines increases the chance of having a recommendation more suited to a particular dataset. Alternatively, one method of expanding on the current findings would be to incorporate elements from Bilalli et al. (2018) and include a rule-based approach into pipeline selection. It would be possible to move from a single meta-dataset to multiple smaller meta-datasets depending on a categorisation of an incoming dataset that requires a DPP pipeline recommendation. Thus, multiple meta-models could be trained, each using different meta-targets to solve specific issues according to rule-based categorisation. This extension would increase the number of possible DPP pipelines that can be recommended and enable the meta-models to focus on more specialised knowledge bases and theoretically improve prediction accuracy.

References

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), 433–459. doi: 10.1002/wics.101

Alam, S., & Yao, N. (2019). The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. *Computational and Mathematical Organization Theory*, *25*, 319–335. doi: 10.1007/s10588-018-9266-8

Alcobaça, E., & Siqueira, F. (2018). *Meta-feature Description Table — pymfe 0.4.1 documentation.* Retrieved 2022-07-27, from `https://pymfe.readthedocs.io/en/latest/auto_pages/meta_\features_description.html`

Alibrahim, H., & Ludwig, S. A. (2021). Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1551–1559). doi: 10.1109/CEC45853.2021.9504761

Anand, K., Wang, Z., Loog, M., & van Gemert, J. (2020). Black Magic in Deep Learning: How Human Skill Impacts Network Training. *CoRR*, *abs/2008.05981*. Retrieved from `https://arxiv.org/abs/2008.05981`

Babii, A., Ghysels, E., & Striaukas, J. (2022). Machine learning time series regressions with an application to nowcasting. *Journal of Business & Economic Statistics*, *40*(3), 1-33. doi: 10.1080/07350015.2021.1899933

Bagui, S., & Li, K. (2021). Resampling imbalanced data for network intrusion detection datasets. *Journal of Big Data*, *8*(1), 1–41. doi: 10.1186/s40537-020-00390-x

Basnet, B., Chun, H., & Bang, J. (2020). An intelligent fault detection model for fault detection in photovoltaic systems. *Journal of Sensors*, *2020*(1), 1–11. doi: 10.1155/2020/6960328

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20–29. doi: 10.1145/1007730.1007735

Ben-Gal, I. (2005). Outlier Detection. *Data Mining and Knowledge Discovery Handbook*, 131–146. doi: 10.1007/0-387-25465-X_7

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, *13*(2), 281-305.

Bilalli, B., Abelló, A., Aluja-Banet, T., & Wrembel, R. (2018). PRESISTANT: Learning based assistant for data pre-processing. *CoRR*, *abs/1803.01024*. Retrieved from `http://arxiv.org/abs/1803.01024`

Brazdil, P., Giraud-Carrier, C., Soares, C., & Vilalta, R. (2009). Metalearning: Concepts and Systems. In *Metalearning: Applications to Data Mining* (pp. 1–10). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-540-73263-1_1

Brownlee, J. (2020). Data Cleaning. In *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python* (pp. 47–51). Machine Learning Mastery, 2020.

Bühlmann, P., & Van De Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media. doi: 10.1007/978-3-642-20192-9

Cerda, P., & Varoquaux, G. (2019). Encoding high-cardinality string categorical variables. *CoRR*, *abs/1907.01860*. Retrieved from `http://arxiv.org/abs/1907.01860`

Chandrasekar, P., & Qian, K. (2016). The impact of data preprocessing on the performance of a Naive Bayes classifier. In *2016 IEEE 40th annual computer software and applications conference (COMPSAC)* (Vol. 2, pp. 618–619). doi: 10.1109/COMPSAC.2016.205

Chatterjee, S., Bopardikar, R., Guerard, M., Thakore, U., & Jiang, X. (2022). MOSPAT: AutoML based Model Selection and Parameter Tuning for Time Series Anomaly Detection. *CoRR*, *abs/2205.11755*. Retrieved from `https://doi.org/10.48550/arXiv.2205.11755`

Chen, B., Wu, H., Mo, W., Chattopadhyay, I., & Lipson, H. (2018). Autostacker: A Compositional Evolutionary Learning System. *CoRR*, *abs/1803.00684*. Retrieved from `http://arxiv.org/abs/1803.00684`

Chen, S., Wu, J., & Liu, X. (2021). EMORL: Effective multi-objective reinforcement learning method for hyperparameter optimization. *Engineering Applications of Artificial Intelligence*, *104*, 104315. doi: 10.1016/j.engappai.2021.104315

Chirita, P.-A., Diederich, J., & Nejdl, W. (2005). MailRank: Using ranking for spam detection. In *International Conference on Information and Knowledge Management, Proceedings* (p. 373-380). doi: 10.1145/1099554.1099671

Chou, J.-S., & Nguyen, T.-K. (2018). Forward forecast of stock price using sliding-window metaheuristic-optimized machine-learning regression. *IEEE Transactions on Industrial Informatics*, *14*(7), 3132–3142. doi: 10.1109/TII.2018.2794389

Crespo Márquez, A. (2022). The Curse of Dimensionality. In A. Crespo Márquez (Ed.), (Vol. 1, pp. 67–86). Springer International Publishing. doi: 10.1007/978-3-030-97660-6_7

Dal Pozzolo, A., Caelen, O., & Bontempi, G. (2015). When is undersampling effective in unbalanced classification tasks? In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15* (pp. 200–215). doi: 10.1007/978-3-319-23528-8_13

Dorfman, R., & Tamar, A. (2020). *Offline Meta Reinforcement Learning* (Vol. abs/2008.02598). Retrieved from `https://arxiv.org/abs/2008.02598`

Duan, L., Xu, L., Liu, Y., & Lee, J. (2009). Cluster-based outlier detection. *Annals of Operations Research*, *168*(1), 151–168. doi: 10.1007/s10479-008-0371-9

Dyrmishi, S., Elshawi, R., & Sakr, S. (2019). A decision support framework for AutoML systems: A meta-learning approach. In *2019 International Conference on Data Mining Workshops (ICDMW)* (pp. 97–106). doi: 10.1109/ICDMW.2019.00025

Elhilbawi, H., Eldawlatly, S., & Mahdi, H. (2021). The importance of discretization methods in machine learning applications: A case study of predicting ICU mortality. In *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2021* (pp. 214–224). doi: 10.1007/978-3-030-69717-4_23

Escalante, H. J. (2020). Automated Machine Learning - a brief review at the end of the early years. *CoRR*, *abs/2008.08516*. Retrieved from `https://arxiv.org/abs/2008.08516`

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, *61*(2), 863–905. doi: 10.1613/jair.1.11192

Feurer, M., Springenberg, J. T., & Hutter, F. (n.d.). Using meta-learning to initialize Bayesian optimization of hyperparameters. *CEUR Workshop Proceedings*, *1201*, 3–10. doi: 10.1609/aaai.v29i1.9354

Fischer, L., Ehrlinger, L., Geist, V., Ramler, R., Sobieczky, F., Zellinger, W., ... Moser, B. (2020). Ai system engineering—key challenges and lessons learned. *Machine Learning and Knowledge Extraction*, *3*(1), 56-83. doi: 10.3390/make3010004

FraunhoferIPT. (2022). *Machine Learning Datasets for Production.* Retrieved 2022-08-01, from `https://www.bigdata-ai.fraunhofer.de/s/datasets/index.html`

Frye, M. (2023). Recommending Data Preprocessing Pipelines for Machine Learning Applications in Production. *Not yet published*.

García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., & Herrera, F. (2016). Big data preprocessing: methods and prospects. *Big Data Analytics*, *1*(1). doi: 10.1186/S41044-016-0014-0

Garouani, M., Ahmad, A., Bouneffa, M., Lewandowski, A., Bourguin, G., & Hamlich, M. (n.d.). Towards the Automation of Industrial Data Science: A Meta-learning based Approach. In *ICEIS (1 doi = 10.5220/0010457107090716, pages=709–716, year=2021.*

Ghosh, A., Satvaya, P., Kundu, P., & Sarkar, G. (2022). Calibration of RGB sensor for estimation of real-time correlated color temperature using machine learning regression techniques. *Optik - International Journal for Light and Electron Optics*, *258*, 168954. doi: 10.1016/j.ijleo.2022.168954

Giovanelli, J., Bilalli, B., & Abelló Gamazo, A. (2021). Effective data pre-processing for AutoML. In *Proceedings of the 23rd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP): co-located with the 24th International Conference on Extending Database Technology and the 24th International Conference on Database Theory (EDBT/ICDT 2021): Nicosia, Cyprus, March 23, 2021* (pp. 1–10).

Griffiths, T. L., Callaway, F., Chang, M. B., Grant, E., Krueger, P. M., & Lieder, F. (2019). Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, *29*, 24-30. doi: https://doi.org/10.1016/j.cobeha.2019.01.005

Gupta, H., & Asha, V. (2020). Impact of Encoding of High Cardinality Categorical Data to Solve Prediction Problems. *Journal of Computational and Theoretical Nanoscience*, *17*(9), 4197–4201. doi: 10.1166/JCTN.2020.9044

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Ho, T., ... Viegas, E. (2015). Design of the 2015 ChaLearn AutoML challenge. In (Vol. 2015-September, pp. 1–8). doi: 10.1109/IJCNN.2015.7280767

Hadwan, M., Al-Sarem, M., Saeed, F., & Al-Hagery, M. A. (2022). An improved sentiment classification approach for measuring user satisfaction toward governmental services' mobile apps using machine learning methods with feature engineering and SMOTE technique. *Applied Sciences*, *12*(11), 5547. doi: 10.3390/app12115547

Haykin, S., & Network, N. (2004). A comprehensive foundation. *Neural networks*, *2*(2004), 41.

He, X., Zhao, K., & Chu, X. (2019). AutoML: A Survey of the State-of-the-Art. *CoRR*, *abs/1908.00709*. Retrieved from `http://arxiv.org/abs/1908.00709`

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and intelligent optimization: 5th international conference, lion 5, rome, italy, january 17-21, 2011. selected papers 5* (pp. 507–523). Berlin, Heidelberg: Springer Berlin Heidelberg.

Jiao, Y., Li, Z., Chen, X., & Fei, S. (2020). Preprocessing methods for near-infrared spectrum calibration. *Journal of Chemometrics*, *34*(11), e3306. doi: 10.1002/CEM.3306

Kano, T., Sakti, S., & Nakamura, S. (2021). Transformer-based direct speech-to-speech translation with transcoder. In *2021 IEEE Spoken Language Technology Workshop (SLT)* (pp. 958–965). doi: 10.1109/SLT48900.2021.9383496

Khalid, S., Khalil, T., & Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. *Proceedings of 2014 Science and Information Conference, SAI 2014*, 372–378. doi: 10.1109/SAI.2014.6918213

Klein, A., Falkner, S., Bartels, S., Hennig, P., & Hutter, F. (2016). Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. *CoRR*, *abs/1605.07079*. Retrieved from `http://arxiv.org/abs/1605.07079`

Komer, B., Bergstra, J., & Eliasmith, C. (2014). Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *ICML workshop on AutoML* (Vol. 9, p. 50). Springer International Publishing. doi: 10.1007/978-3-030-05318-5_5

Krouwer, J. S., & Schlain, B. (1993). A method to quantify deviations from assay linearity. *Clinical Chemistry*, *39*(8), 1689–1693. doi: 10.1093/clinchem/39.8.1689

Laadan, D., Vainshtein, R., Curiel, Y., Katz, G., & Rokach, L. (2019). Rankml: a meta learning-based approach for pre-ranking machine learning pipelines. *CoRR*, *abs/1911.00108*. Retrieved from `http://arxiv.org/abs/1911.00108`

Lavangnananda, K., & Chattanachot, S. (2017). Study of discretization methods in classification. In *2017 9th International Conference on Knowledge and Smart Technology (KST)* (pp. 50–55). doi: 10.1109/KST.2017.7886082

Lee, J. W., & Giraud-Carrier, C. (2014). On the dangers of default implementations: The case of radial basis function networks. *Intelligent Data Analysis*, *18*(2), 261–279. doi: 10.3233/IDA-140640

Li, C. (2019). Preprocessing Methods and Pipelines of Data Mining: An Overview. *CoRR*, *abs/1906.08510*. Retrieved from `http://arxiv.org/abs/1906.08510`

Lin, W. C., & Tsai, C. F. (2019). Missing value imputation: a review and analysis of the literature (2006–2017). *Artificial Intelligence Review 2019*, *53*(2), 1487–1509. doi: 10.1007/S10462-019-09709-4

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, *6*(4), 393–423. doi: 10.1023/A:1016304305535

Liu, Z., Xu, Z., Madadi, M., Junior, J. J., Escalera, S., Rajaa, S., & Guyon, I. (2019). Overview and unifying conceptualization of Automated Machine Learning. In *Proceedings of the automating data science workshop, wurzburg, germany* (Vol. 20). Wurzburg, Germany: Proc. Automat. Data Sci. Workshop.

Longadge, R., & Dongre, S. (2013). Class Imbalance Problem in Data Mining Review. *CoRR*, *abs/1305.1707*. Retrieved from `http://arxiv.org/abs/1305.1707`

Lu, C.-P., Liaw, J.-J., Wu, T.-C., & Hung, T.-F. (2019). Development of a mushroom growth measurement system applying deep learning for image recognition. *Agronomy*, *9*(1), 32. doi: 10.3390/agronomy9010032

McGinnis, W., & Westenthanner, P. (2015). category_encoders. Retrieved from `https://github.com/scikit-learn-contrib/category_encoders`

Munson, M. A. (2012). A study on the importance of and time spent on different modeling steps. *ACM SIGKDD Explorations Newsletter*, *13*(2), 65–71. doi: 10.1145/2207243.2207253

Nagarajah, T., & Poravi, G. (2019). A Review on Automated Machine Learning (AutoML) Systems. In *2019 IEEE 5th International Conference for Convergence in Technology, I2CT 2019.* Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/I2CT45611.2019.9033810

Napoleon, D., & Pavalakodi, S. (2011). A new method for dimensionality reduction using k-means clustering algorithm for high dimensional data set. *International Journal of Computer Applications*, *13*(7), 41–46. doi: 10.5120/1789-2471

Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Kidd, L. C., & Moore, J. H. (2016). Automating biomedical data science through tree-based pipeline optimization. *CoRR*, *abs/1601.07925*. Retrieved from `http://arxiv.org/abs/1601.07925`

Peng, Y., Flach, P. A., Soares, C., & Brazdil, P. (2002). Improved dataset characterisation for meta-learning. In *Discovery Science: 5th International Conference, DS 2002 Lübeck, Germany, November 24–26, 2002 Proceedings 5* (pp. 141–152). doi: 10.1007/3-540-36182-0_14

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-Learning by Landmarking Various Learning Algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning ICML2000* (Vol. 951, pp. 743–750). Morgan Kaufmann Publishers Inc. doi: 10.5555/645529.658105

A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. (2001). *ACM SIGKDD Explorations Newsletter*, *3*(1), 27–32. doi: 10.1145/507533.507538

Quemy, A. (2019). Data pipeline selection and optimization. In *CEUR Workshop Proceedings* (Vol. 2324).

Rajendran, J., Irpan, A., & Jang, E. (2020). Meta-Learning Requires Meta-Augmentation. *CoRR*, *abs/2007.05549*. Retrieved from `https://arxiv.org/abs/2007.05549`

Refaeilzadeh, P., Tang, L., Liu, H., et al. (2009). Cross-validation. *Encyclopedia of database systems*, *5*, 532–538. doi: 10.1007/978-0-387-39940-9_565

Rice, J. R. (1976). The Algorithm Selection Problem. *Advances in Computers*, *15*(C), 65–118. doi: 10.1016/S0065-2458(08)60520-3

Rivolli, A., Garcia, L. P., Soares, C., Vanschoren, J., & de Carvalho, A. C. (2022). Meta-features for meta-learning. *Knowledge-Based Systems*, *240*, 108101. doi: 10.1016/j.knosys.2021.108101.

Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. *Degree Project Technology 2018:596*, 41.

Sharma, V. (2022). A Study on Data Scaling Methods for Machine Learning. *International Journal for Global Academic & Scientific Research*, *1*(1), 23–33. doi: 10.55938/ijgasr.v1i1.4

Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, *5*(4), 13–23.

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*, 105524. doi: 10.1016/J.ASOC.2019.105524

Steinbach, M., Ertöz, L., & Kumar, V. (2004). The Challenges of Clustering High Dimensional Data. In L. T. Wille (Ed.), (pp. 273–309). Berlin, Heidelberg: Springer, Berlin, Heidelberg. doi: 10.1007/978-3-662-08968-2_16

Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K.-R. (2021). Towards CRISP-ML (Q): a machine learning process model with quality assurance methodology. *Machine learning and knowledge extraction*, *3*(2), 392–413. doi: 10.3390/make3020020

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2012). Auto-WEKA: Automated Selection and Hyper-Parameter Optimization of Classification Algorithms. In (Vol. abs/1208.3719). Retrieved from `http://arxiv.org/abs/1208.3719`

Vanschoren, J. (2019). Meta-Learning. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated Machine Learning* (1st ed., pp. 35–61). Cham: Springer Nature. doi: 10.1007/978-3-030-05318-5_2

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2014). OpenML: networked science in machine learning. *CoRR*, *abs/1407.7722*. Retrieved from `http://arxiv.org/abs/1407.7722`

Wistuba, M., Rawat, A., & Pedapati, T. (2019). A Survey on Neural Architecture Search. *CoRR*, *abs/1905.01392*. Retrieved from `http://arxiv.org/abs/1905.01392`

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., & Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, *17*(1), 26-40. doi: 10.11989/JEST.1674-862X.80904120

Xanthopoulos, P., Pardalos, P. M., Trafalis, T. B., Xanthopoulos, P., Pardalos, P. M., & Trafalis, T. B. (2013). Linear discriminant analysis. *Robust data mining*, 27–33. doi: 10.1007/978-1-4419-9878-1_4

Yang, C., Akimoto, Y., Kim, D. W., & Udell, M. (2018). OBOE: Collaborative Filtering for AutoML Initialization. *CoRR*, *abs/1808.03233*. Retrieved from `http://arxiv.org/abs/1808.03233`

Yang, C., Fan, J., Wu, Z., & Udell, M. (2020). AutoML Pipeline Selection: Efficiently Navigating the Combinatorial Space. In (p. 1446-1456). doi: 10.1145/3394486.3403197

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *CoRR*, *abs/2007.15745*. Retrieved from `https://arxiv.org/abs/2007.15745`

Yao, H., Huang, L., Wei, Y., Tian, L., Huang, J., & Li, Z. (2020). Don't Overlook the Support Set: Towards Improving Generalization in Meta-learning. *CoRR*, *abs/2007.13040*. Retrieved from `https://arxiv.org/abs/2007.13040`

Zagatti, F. R. (2021). MetaPrep: Data preparation pipelines recommendation via meta-learning. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA).* doi: 10.1109/ICMLA52953.2021.00194

Appendices

A Training Set Overview

Names and description of datasets used to train the meta model. Many of the datasets were separated into datasets prediction different target values. List obtained from FraunhoferIPT (2022)

Name	Published	Description
3D Printer	2018	Dataset aimed toward estimating how adjustment parameters in 3d printers affect print quality, accuracy and strength.
Airfoil Self-Noise	2014	The NASA data set comprises different size NACA 0012 airfoils at various wind tunnel speeds and angles of attack. The goal is to predict sound pressure levels.
APS Scania Trucks	2018	Data from heavy Scania trucks in daily usage. Focuses on the Air Pressure system (APS) which generates pressurised air used in various functions, such as braking and gear shifting.
Bolts	2014	Data from an experiment on the effects of machine adjustments on the time to count bolts. Bolts are dumped into a large metal dish.
CNC Milling	2018	Machining data was collected from a CNC machine for variations of tool condition, feed rate, and clamping pressure.
Condition Monitoring Hydraulic Systems	2018	The data set addresses the condition assessment of a hydraulic test rig based on multi-sensor data. Four fault types are superimposed with several severity grades impeding selective quantification.
Cylinder Bands	1995	Process delays known as cylinder banding in rotogravure printing were substantially mitigated using control rules.
Degradation of a Cutting Blade	2019	Wrapping machine process data over 12 months with a degrading cutting tool.
Flight Software for Earth Orbiting Satellite	2014	One of the NASA Metrics Data Program defect data sets. Data from flight software for earth orbiting satellite.
Genesis Demonstrator	2018	The Genesis Demonstrator is a portable pick-and-place demonstrator which uses an air tank to supply gripping and storage units.
High Storage System Energy Optimisation	2018	Data collected from a demonstrator of a high storage system. The high storage system consists of 4 short conveyor belts and 2 rails.
Laser Welding	2020	Process parameter recordings for correlation with weld quality indicators such as weld depth and geometrical dimensions.
Maintenance of Naval Propulsion Plants	2014	Data generated from a sophisticated simulator of a Gas Turbines (GT), mounted on a Frigate in a gas propulsion plant.
Mechanical Analysis	1990	Fault diagnosis problem of electromechanical devices. Each instance contains many components, each of which has 8 attributes. Different instances in this database have different numbers of components.
Mercedes-Benz Greener Manufacturing	2016	This dataset contains an anonymised set of variables, each representing a custom feature in a Mercedes car.
Milling	2007	Experiments on a milling machine for different speeds, feeds, and depth of cut.
Pulsar Star	2018	A dataset consisting of pulsar candidates collected during the High Time Resolution Universe Survey
Robot Execution Failures	1999	This dataset contains force and torque measurements on a robot after failure detection.
SECOM	2008	A complex modern semi-conductor manufacturing process is normally under consistent surveillance via the monitoring of signals collected from sensors.
Steel Plates Faults	2017	A data set of steel plates faults, classified into 7 different types.
Software for Ground Data	2014	NASA Metrics Data Program defect data sets: Data from software for storage management for receiving and processing ground data.
Turbofan Engine Degradation Simulation	2020	Run-to-failure trajectories for a small fleet of aircraft engines under realistic flight conditions.

B Meta-Feature List

List of meta-features taken from the pymfe library written by (Alcobaça & Siqueira, 2018). Note: two meta-features were included for all meta-features denoted by *. For these measures both the average value and the standard deviation were used as meta-features.

Group	Name	Description
General	nr_inst	Compute the number of instances (rows) in the dataset.
	attr_to_inst	Compute the ratio between the number of attributes.
	inst_to_attr	Compute the ratio between the number of instances and attributes.
	nr_attr	Compute the total number of attributes.
	nr_bin	Compute the number of binary attributes.
	freq_class*	Compute the relative frequency of each distinct class.
	nr_num	Compute the number of numeric features.
	nr_class	Compute the number of distinct classes.
Info-Theory	mut_inf*	Compute the mutual information between each attribute and target.
	eq_num_attr	Compute the number of attributes equivalent for a predictive task.
	attr_conc*	Compute concentration coef. of each pair of distinct attributes.
	ns_ratio	Compute the noisiness of attributes.
	class_ent	Compute target attribute Shannon's entropy.
	class_conc*	Compute concentration coefficient between each attribute and class.
	attr_ent*	Compute Shannon's entropy for each predictive attribute.
	joint_ent*	Compute the joint entropy between each attribute and class.
Landmarking	one_nn*	Performance of the 1-Nearest Neighbor classifier.
	naive_bayes*	Performance of the Naive Bayes classifier.
	random_node*	Performance of the single decision tree node model induced by a random attribute.
	linear_discr*	Performance of the Linear Discriminant classifier.
	worst_node*	Performance of the single decision tree node model induced by the worst informative attribute.
	best_node*	Performance of a the best single decision tree node.
	elite_nn*	Performance of Elite Nearest Neighbor.

Group	Name	Description
Statistical	kurtosis*	Compute the kurtosis of each attribute.
	var*	Compute the variance of each attribute.
	t_mean*	Compute the trimmed mean of each attribute.
	sparsity*	Compute (possibly normalized) sparsity metric for each attribute.
	skewness*	Compute the skewness for each attribute.
	sd_ratio	Compute a statistical test for homogeneity of covariances.
	sd*	Compute the standard deviation of each attribute.
	roy_root	Compute the Roy's largest root.
	range*	Compute the range (max - min) of each attribute.
	p_trace	Compute the Pillai's trace.
	can_cor*	Compute canonical correlations of data.
	cor*	Compute the absolute value of the correlation of distinct dataset column pairs.
	nr_outliers	Compute the number of attributes with at least one outlier value.
	cov*	Compute the absolute value of the covariance of distinct dataset attribute pairs.
	nr_norm	Compute the number of attributes normally distributed based in a given method.
	eigenvalues*	Compute the eigenvalues of covariance matrix from dataset.
	nr_disc	Compute the number of canonical correlation between each attribute and class.
	nr_cor_attr	Compute the number of distinct highly correlated pair of attributes.
	g_mean*	Compute the geometric mean of each attribute.
	gravity	Compute the distance between minority and majority classes center of mass.
	h_mean*	Compute the harmonic mean of each attribute.
	iq_range*	Compute the interquartile range (IQR) of each attribute.
	min*	Compute the minimum value from each attribute.
	w_lambda	Compute the Wilks' Lambda value.
	mean*	Compute the mean value of each attribute.
	max*	Compute the maximum value from each attribute.
	mad*	Compute the Median Absolute Deviation (MAD) adjusted by a factor.
	lh_trace	Compute the Lawley-Hotelling trace.
	median*	Compute the median value from each attribute.

C Test Set Overview

Names and description of datasets used to test the meta model. Many of the datasets were separated into datasets prediction different target values. Each dataset was obtained from the website named in the ”Loaction” column.

Name	Location	Description
Civil Engineering: Cement Manufacturing Dataset	Kaggle	The actual concrete compressive strength for a given mixture under a specific age
Electrochemical Ammonia Removal and Disinfection of Aquaculture Wastewater	Havard Dataverse	electrochemical ammonia removal and disinfection of wastewater from an aquaculture farm in Hawaii.
Image Segmentation	Sci2s	The dataset describes main characteristics of visual image segmentation data with an imbalanced class distribution
Membrane Electrode Assembly Activation Procedures	Fuel Cells Etc.	This dataset are about Nafion 112 membrane standard tests and MEA activation tests of PEM fuel cell in various operation condition.
National Material Capabilities	Data.world	The National Material Capabilities data set contains annual values for total population, urban population, iron and steel production, energy consumption, military personnel, and military expenditure of all state members, currently from 1816-2007
Single DBFC Dataset	Kaggle	This dataset includes Direct Borohydride Fuel Cell (DBFC) impedance and polarization test in anode with Pd/C, Pt/C and Pd decorated Ni–Co/rGO catalysts.
Used Cars Dataset	Kaggle	Cars' data was scraped from tc-v.com and it included Information about Japan's largest online used car marketplace.
Weather Buoy Network	Data.world	Real time meteorological and oceanographic data collected from the Irish moored Weather Buoy network of stations.

D Self Implemented DPP Methods

Algorithm 1: ZScore Substitution

```
Input: $X \leftarrow DataFrame$
Output: X
1 $X_{ZScores} \leftarrow CalculateZScores(X)$
2 for $Column$ in $X_{ZScores}$ do
3     $ColumnMean = Mean(Column)$
4     for $ZScoreValue$ in $Column$ do
5         if $ZScoreValue >= 3$ then
6             $X[ColumnIndex, ValueIndex] \leftarrow ColumnMean$
7
```

Algorithm 2: Random Noise Addition

```
Input: $X \leftarrow DataFrame$
Output: $X_{Noise}$
1 $\mu \leftarrow 0$
2 $\sigma \leftarrow 0.1$
3 $Noise \leftarrow GenerateRandomNoise(\mu, \sigma, Shape(X))$
4 $NotSelectedColumns \leftarrow SelectPercentageOfColumns(Noise, 20\%)$
5 $X[CategoricalColumns] \leftarrow 0$
6 $Noise[NotSelectedColumns] \leftarrow 0$
7 $X_{Noise} \leftarrow X + Noise$
```

Algorithm 3: Remove Correlated Features

```
Input: $X \leftarrow DataFrame$
Output: X
1 $SelectedColumns \leftarrow GetColumns(X)$
2 $threshold = 0.8$
3 $X_{CorrelationMatrix} \leftarrow GetCorrelationMatrix(X)$
4 for $Column$ in $X_{CorrelationMatrix}$ do
5     for $CorrelationValue$ in $Column$ do
6         if $CorrelationValue >= threshold$ then
7             $SelectedColumns \leftarrow SelectedColumns.Remove(Column)$
8 $X \leftarrow X[SelectedColumns]$
9
```

Algorithm 4: Dummy Method

```
Input: $X \leftarrow DataFrame$
Output: $X$
```

E Dataset Names

List of dataset names according to x-axis of Figure 10

Classification Datasets 1	Classification Datasets 2	Regression Datasets
0: Mechanical Analysis	37: Degredation NewBlade Cl mvs	0: Turbine
1: Robot Execution Failures	38: Unknown2 both	1: Compressor
2: Accumulator both	39: Degredation NewBlade Cl both	2: Turbine mvs
3: Accumulator mvs	40: Degredation NewBlade Cl cats	3: Compressor mvs
4: Accumulator cats	41: Unknown2	4: Turbine both
5: Accumulator	42: Degredation NewBlade Cl	5: Compressor both
6: FSEOS(1)	43: Unknown2 cats	6: Degredation NewBlade Re
7: Secom	44: EOAO	7: Degredation NewBlade Re mvs
8: SECOM mvs	45: EOAS	8: Turbine cats
9: FSEOS(1) mvs	46: GDMS	9: Degredation NewBlade Re both
10: SFGD cats	47: Cylinder Bands	10: Compressor cats
11: SFGD	48: CNC Mill Tool Wear	11: Degredation NewBlade Re cats
12: SECOM both	49: APS	12: Milling
13: SECOM cats	50: GDAD both	13: Airfoil Self-Noise SPL
14: Pump mvs	51: Flag cats	14: Airfoil Self-Noise SPL mvs
15: FSEOS(1) cats	52: Flag both	15: Milling Spindle
16: Pump	53: Flag mvs	16: Milling Table
17: FSEOS(1) both	54: Pulsar Star cats	17: Airfoil Self-Noise
18: SFGD both	55: Pulsar Star both	18: Airfoil Self-Noise mvs
19: Pump both	56: Flag	19: Airfoil Self-Noise SPL cats
20: SFGD mvs	57: Pulsar Star mvs	20: 3D Printer
21: Unknown1_MT	58: Pulsar Star	21: Airfoil Self-Noise SPL both
22: Unknown1	59: GDAD mvs	22: 3D Printer Elongation
23: Unknown2_MT	60: Cooler	23: Airfoil Self-Noise cats
24: FSEOS mvs	61: Laser Welding	24: Airfoil Self-Noise both
25: FSEOS	62: Valve mvs	25: Laser Welding Depth
26: Unknown1 cats	63: Valve both	26: Turbo Fan 2
27: Steel Plates Faults mvs	64: GDAD	27: Turbo Fan 4
28: FSEOS cats	65: GDAD cats	28: 3D Printer Strength
29: Pump cats	66: Valve cats	29: Mercedes
30: Unknown1 both	67: Valve	30: Turbo Fan 3
31: FSEOS both	68: GDM	31: Turbo Fan 1
32: Steel Plates Faults both	69: Bolts both	32: 3D Printer Roughness
33: Steel Plates Faults cats	70: Bolts cats	
34: Steel Plates Faults	71: Bolts	
35: Unknown1 mvs	72: Bolts mvs	
36: Unknown2 mvs		

F Chosen Pipelines

Pipelines and pipeline median rank over all usecases of selected pipeline pool for classification datasets

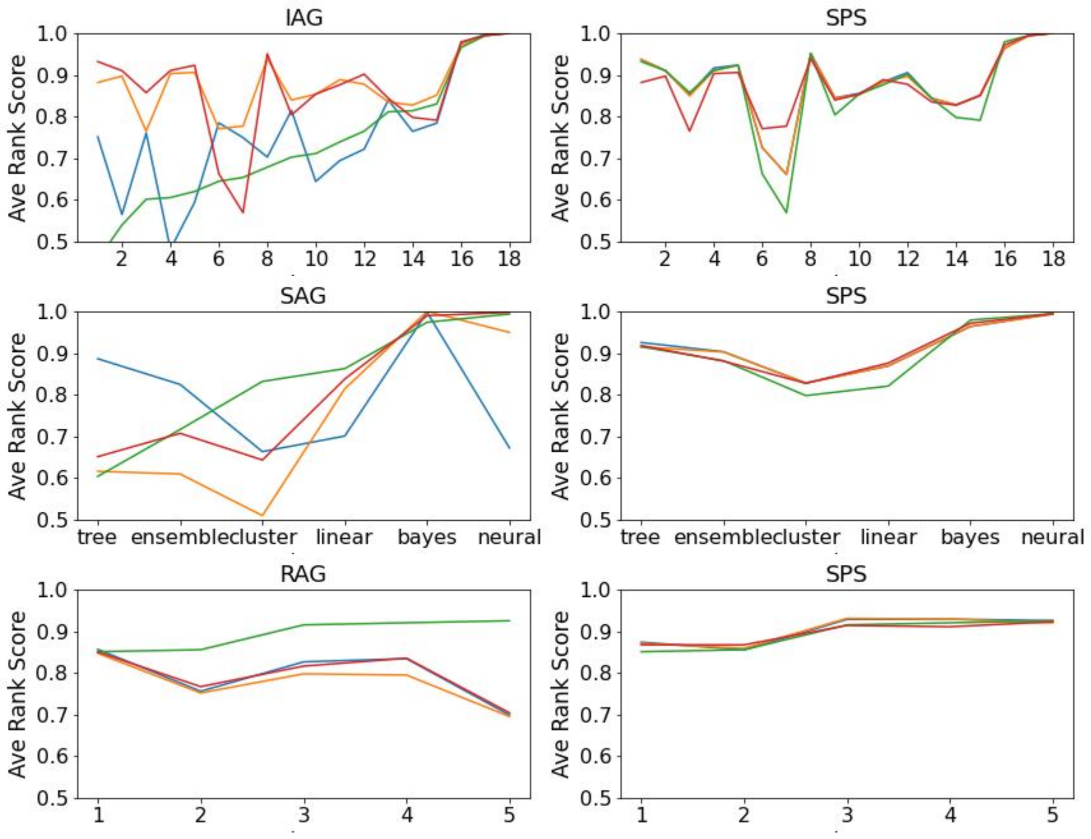
Selection Method	DPP Pipeline	Median Rank	Average
IAG	LastObservedImputer, OneHotEncoder, ZScoreSubstitution, StandardScaler, PCA, RemoveCorrelated	0.71012	0.7341
	MedianImputer, OneHotEncoder, ZScoreSubstitution, RandNoise	0.7082	
	MeanImputer, OneHotEncoder, StandardScaler, PowerTransformer, SMOTE	0.74914	
	MeanImputer, OneHotEncoder, RandNoise, SMOTE	0.76892	
SAG	MeanImputer, OneHotEncoder,ZScoreSubstitution, StandardScaler, PowerTransformer, PCA, RemoveCorrelated, RandNoise	0.7361	0.7356
	MedianImputer, OneHotEncoder, ZScoreSubstitution, RandNoise	0.7136	
	MeanImputer, OneHotEncoder, MinMaxScaler, PowerTransformer	0.76199	
	MeanImputer, OneHotEncoder, StandardScaler, PowerTransformer, SMOTE	0.73071	
RAG	MeanImputer, OneHotEncoder, StandardScaler, RandNoise, SMOTE	0.80105	0.7838
	MeanImputer, OneHotEncoder, ZScoreSubstitution, StandardScaler, VarianceThreshold	0.78886	
	MedianImputer, OneHotEncoder, MinMaxScaler, RandNoise	0.76012	
	MeanImputer, OneHotEncoder, StandardScaler, SMOTE	0.78518	
SPS	MeanImputer, OneHotEncoder, ZScoreSubstitution, StandardScaler, RandNoise, SMOTE	0.85215	0.85016
	MedianImputer, OneHotEncoder, ZScoreSubstitution, StandardScaler, VarianceThreshold, RandNoise, SMOTE	0.85187	
	MeanImputer, OneHotEncoder, StandardScaler, RandNoise, SMOTE	0.84913	
	MeanImputer, OneHotEncoder, StandardScaler, SMOTE	0.84748	

Pipelines and pipeline median rank over all usecases of selected pipeline pool for regression datasets

Selection Method	DPP Pipeline	Median Rank	Average
IAG	MedianImputer, OneHotEncoder, StandardScaler, PowerTransformer, RemoveCorrelated, RandNoise	0.75587	0.80782
	MedianImputer, OneHotEncoder, StandardScaler, RandNoise	0.87953	
	MedianImputer, OneHotEncoder, PowerTransformer, RandNoise	0.70696	
	MedianImputer, OneHotEncoder, ZScoreSubstitution, MinMaxScaler	0.8889	
SAG	MedianImputer, OneHotEncoder, ZScoreSubstitution, RandNoise	0.76277	0.7744
	MedianImputer, TargetEncoder, ZScoreSubstitution, RemoveCorrelated, RandNoise	0.71521	
	MedianImputer, OneHotEncoder, ZScoreSubstitution, PowerTransformer	0.84719	
	MeanImputer, OneHotEncoder, ZScoreSubstitution, MinMaxScaler, RemoveCorrelated, RandNoise	0.77243	
RAG	MedianImputer, OneHotEncoder, ZScoreSubstitution, RandNoise	0.83339	0.87724
	MedianImputer, OneHotEncoder, StandardScaler, RandNoise	0.87778	
	MedianImputer, TargetEncoder, ZScoreSubstitution, StandardScaler	0.89848	
	MedianImputer, TargetEncoder, ZScoreSubstitution, StandardScaler, VarianceThreshold	0.8993	
SPS	MedianImputer, TargetEncoder, ZScoreSubstitution, StandardScaler, VarianceThreshold	0.91312	0.90587
	MedianImputer, TargetEncoder, ZScoreSubstitution, StandardScaler	0.9122	
	MedianImputer, OneHotEncoder, ZScoreSubstitution, MinMaxScaler	0.90673	
	MedianImputer, OneHotEncoder, StandardScaler, RandNoise	0.89144	

G Selected Pipelines for Regression Usecases

Performance of pipelines selected by the different pipeline selectors (IAG, SAG, RAG) for the groups specified by the individual pipeline selector for regression usecases as well as the performances of the SPS pipelines as a comparison.



H Cross Validation Folds CL

Overview of the classification datasets within the meta datasets that were used to define the folds for cross validation

fold 1	fold 2	fold 3	fold 4
Accumulator	EOAO	Flag	Secom
Accumulator both	EOAS	Flag both	SECOM both
Accumulator cats	FSEOS	Flag cats	SECOM cats
Accumulator mvs	FSEOS both	Flag mvs	SECOM mvs
Degredation NewBlade Cl	FSEOS cats	GDAD	Pump
Degredation NewBlade Cl both	FSEOS mvs	GDAD both	Pump both
Degredation NewBlade Cl cats	FSEOS(1)	GDAD cats	Pump cats
Degredation NewBlade Cl mvs	FSEOS(1) both	GDAD mvs	Pump mvs
	FSEOS(1) cats	GDM	Cooler
	FSEOS(1) mvs	GDMS	Cylinder Bands
		APS	

fold 5	fold 6	fold 7	
SFGD	Pulsar Star	Unknown1	
SFGD both	Pulsar Star both	Unknown1 both	
SFGD cats	Pulsar Star cats	Unknown1 cats	
SFGD mvs	Pulsar Star mvs	Unknown1 mvs	
Laser Welding	Steel Plates Faults	Unknown1_MT	
Robot Execution Failures	Steel Plates Faults cats	Unknown2 both	
Valve	Steel Plates Faults mvs	Unknown2 cats	
Valve both	Bolts	Unknown2 mvs	
Valve cats	Bolts both	Unknown2_MT	
Valve mvs	Bolts mvs		
	Bolts cats		
	CNC Mill Tool Wear		

I Cross Validation Folds RE

Overview of the classification datasets within the meta datasets that were used to define the folds for cross validation

fold 1	fold 2	fold 3	fold 4
3D Printer	Airfoil Self-Noise	Compressor	Turbo Fan 1
3D Printer Elongation	Airfoil Self-Noise SPL	Compressor both	Turbo Fan 2
3D Printer Roughness	Airfoil Self-Noise SPL mvs	Compressor cats	Turbo Fan 3
3D Printer Strength	Airfoil Self-Noise mvs	Compressor mvs	Turbo Fan 4
Laser Welding Depth	Airfoil Self-Noise both	Degredation NewBlade Re	Milling
Mercedes	Airfoil Self-Noise cats	Degredation NewBlade Re cats	Milling Spindle
Turbine	Airfoil Self-Noise SPL cats	Degredation NewBlade Re both	Milling Table
Turbine both	Airfoil Self-Noise SPL both	Degredation NewBlade Re mvs	
Turbine cats			
Turbine mvs			

J Cross-Validation Results

Best training results for each performance measurement, pipeline selection method and ML algorithm. The results suggest that diverse pipelines produce better results when the ordering of pipelines is required (see F1-Score and $a\tau$), and reliable pipelines produce better ARMSE scores.

Measurement	ML Task	Selector	KNN	Decision Tree	Random Forest	Best
ARMSE	Classification	SAG	0.24505	0.24303	0.23456	0.23456
		IAG	0.26165	0.24793	0.2379	0.2379
		RAG	0.21142	0.21171	0.20136	0.20136
		SPS	0.18956	0.18378	**0.17994**	**0.17994**
	Regression	SAG	0.28856	0.30235	0.27481	0.27481
		IAG	0.25376	0.25252	0.24003	0.24003
		RAG	0.24855	0.23797	0.2421	0.23797
		SPS	0.20387	0.20244	**0.19732**	**0.19732**
$a\tau$	Classification	**SAG**	0.09137	**0.15166**	0.11547	**0.15166**
		IAG	0.04998	0.1166	0.08015	0.1166
		RAG	0.12565	0.10399	0.13404	0.13404
		SPS	0.05227	0.11241	0.1151	0.1151
	Regression	SAG	0.16722	0.14019	0.13448	0.16722
		IAG	0.12704	**0.29487**	0.28788	**0.29487**
		RAG	0.12385	0.15421	0.118	0.15421
		SPS	0.09421	0.15419	0.1058	0.15419
F1-Score	Classification	**SAG**	0.29153	**0.31945**	0.24297	**0.31945**
		IAG	0.25935	0.29945	0.25994	0.29945
		RAG	0.29776	0.27733	0.2514	0.29776
		SPS	0.27148	0.29277	0.26693	0.29277
	Regression	SAG	0.34753	0.30085	0.17588	0.34753
		IAG	0.34291	**0.3493**	0.23836	**0.3493**
		RAG	0.28132	0.33943	0.1515	0.33943
		SPS	0.27566	0.32275	0.27428	0.32275

K Final Model Settings

Overview of the optimal settings determined through crossvalidation for each of the different final meta models. These cover the models for the classification and regression meta models and the three performance measurements

Measurement	ML Task	Algorithm	Parameters	DPP pipeline
ARMSE	CL	RF	'criterion': 'absolute_error', 'min_samples_split': 3, 'min_samples_leaf': 1	PCA, StandardScaler, SMOTE
	RE	RF	'criterion': 'criterion': 'squared_error', 'min_samples_split': 2, 'min_samples_leaf': 1	PCA, StandardScaler, RandNoise
$a\tau$	CL	DT	'criterion': 'squared_error', 'min_samples_split': 4, 'min_samples_leaf': 3	PCA, MinMaxScaler, PowerTransformer
	RE	DT	'criterion': 'squared_error', 'min_samples_split': 3, 'min_samples_leaf': 1	PCA, PowerTransformer, RandNoise
F1-Score	CL	DT	'criterion': 'squared_error', 'min_samples_split': 4, 'min_samples_leaf': 3	PCA, MinMaxScaler, PowerTransformer
	RE	DT	'criterion': 'squared_error', 'min_samples_split': 2, 'min_samples_leaf': 2	PCA, PowerTransformer, RandNoise

L TPOT DPP Methods

List of all DPP methods available for the TPOT optimisation.

DPP Method	Method Hyperparameter	Values
Binariser	Threshold	0.0,...,1.0
Fast ICA	Tol	0.0,...,1.0
Feature Agglomeration	Linkage	['ward', 'complete', 'average']
	Affinity	['euclidean', 'l1', 'l2', 'manhattan', 'cosine']
Max-Abs Scaler		
Min-Abs Scaler		
Normaliser	Norm	['l1', 'l2', 'max']
Nystroem	Kernel	['rbf', 'cosine', 'chi2', 'laplacian', 'polynomial', 'poly', 'linear', 'additive chi2', 'sigmoid']
	Gamma	0.0,...,1.0
	Num Components	1,...,10
PCA	Svd Solver	['randomised']
	Iterated Power	1,...,10
Polynomial Features	Degree	2
	Include Bias	False
	Interaction Only	False
RBF Sampler	Gamma	0.0,...,1.0
Robust Scaler		
Standard Scaler		
Zero Count		
One-Hot Encoder	Minimum Fraction	[0.05, 0.1, 0.15, 0.2, 0.25]
	Sparse	False