1 Operators

In this section, we present a list of all operators which have been implemented and evaluated in GELFE. For each of these operators, all feature sets from the collection of data sets for which the operator can be applied will be tested with that operator to create a meta-target variable for each feature set.

The list of operators can be split into unary and binary operators which, of course, depends on the number of features the operator is applied to. As explained in the paper, operators with an arity larger than two will not be taken into account. An additional split can be made between operators for numerical or categorical features or different combinations of these feature types. As in the collection of data sets, the only distinction was made between numerical and categorical features, it is not possible to create operators for different types of features such as date types. When such a feature is present in a new data set, it should beforehand be converted to appropriate numerical or categorical features as that was probably also done for the online available data sets. Nominal and ordinal features are thus both labeled as categorical features and further distinctions cannot be made. In the next paragraphs, we will enumerate all operators along with several formulas for more clarification and the motivation to use them.

Unary operators. For each operator t_j, f_i is the feature on which the operator is applied with index l and x^* is the resulting feature set. This is denoted as a feature set because some operators return more than one new feature.

Numerical operators. Most numerical operators focus on altering the distribution of a feature, for example when a feature is skewed or contains outliers. Other operators focus on binning the feature to a useful categorical one. The following fifteen numerical operators, which are abbreviated with the N of numerical and a number, have been implemented:

N1: The natural logarithm operator. This is one of the most used numerical operators. Together with the next three operators (the inverse, the square root, and the inverse square root operator) they can all be used to reduce right skewness of a feature. The natural logarithm is a stronger operator than the square root operator. Like many other operators, logarithms cannot handle negative values and will therefore not be applied to features which contain negative values. In other implementations, this problem was solved by taking the absolute value of the complete feature. However, we believe that when a feature contains negative values, information will be lost if the absolute value is taken. Furthermore, the number of features with negative values is not a large part of the features in the complete collection. In Equation 1, the formula to create x^* can be found. If a feature contains a zero, one is added to the complete feature as the logarithm of zero is negative infinity.

$$x_l^* = \begin{cases} \ln(f_{il} + 1), & 0 \in f_i \\ \ln(f_{il}), & \text{otherwise} \end{cases} \tag{1}$$

N2: The inverse operator. This operator, also sometimes called the reciprocal, is one divided by the feature values. This is the strongest operator of the first four operators based on the change to the distribution of the feature. Inverse operators also change the order of the feature

as the maximum value becomes the minimum value of the new feature, given that the feature values are all positive or all negative. This only impacts the interpretation of the new feature such that it is not necessary to correct for this. When the feature contains both positive and negative values, applying this operator has no use as the smallest positive and smallest negative value, which are two consecutive values, become the maximum and minimum value of the new feature after applying the inverse operator. In the implementation, when the feature value f_{il} is zero, a small fraction ϵ should be added to that value in order to not divide by zero. In the implementation, epsilon has been chosen as 0.0001 when all feature values are positive and -0.0001 when all values are negative.

$$x_l^* = \begin{cases} (f_{il} + \epsilon)^{-1}, & f_{il} = 0 \\ (f_{il})^{-1}, & \text{otherwise} \end{cases} \tag{2}$$

N3: The square root operator. This operator is the weakest of the four operators used to reduce right skewness. The square root operator can also not be applied to negative values.

N4: The inverse square root operator. This is a combination of the previous two operators and is used as a less strong operator than the inverse operator. The same trick with the small value ϵ is necessary when the feature value f_{il} is zero and the operator cannot be applied to features containing negative values due to the square root.

$$x_l^* = \begin{cases} (\sqrt{f_{il}} + \epsilon)^{-1}, & f_{il} = 0 \\ (\sqrt{f_{il}})^{-1}, & \text{otherwise} \end{cases} \tag{3}$$

N5: The square operator. The square and the cube operator are mainly used to reduce left skewness.

N6: The cube operator. An advantage of the cube operator compared to the square operator is that negative feature values stay negative.

N7: The robust standardizer. This operator standardizes or scales the feature using the median and the interquartile range. The robust standardizer is implemented because the normal standardizer, which uses the mean and standard deviation, can often be influenced by outliers.

N8: The arctangens operator. This is an operator which enlarges the difference between feature values around zero and flattens the values in the tails of the distribution. It can be said that the function follows an S-shape. Similar operators are the tanh and the logistic function. For the arctangens operator, all values in the tail will be converted to approximately $-\pi/2$ or $\pi/2$ such that outliers will not have a large impact after the operator is used on a feature. Before this operator can be applied, it should first be robustly standardized around zero.

N9: The quantile operator. This operator does only take the order of the values into account and does not look at the distribution of the feature. It transforms the feature to a numeric feature which resembles a uniform distribution. The smallest value will become zero and the largest value one. All other values depend on their order and the number of observations. When the feature contains 101 observations and 101 distinct values, the new feature will contain the values $\{0.00, 0.01, \ldots, 0.99, 1.00\}$. With 11 or 1001 observations, the second observation in order will get the transformed value 0.1 or 0.001.

N10: The equal frequency discretizer with 5 bins. All remaining numerical operators transform the numeric feature to a discrete one. The equal frequency discretizer with k bins splits the feature in k parts which all contain the same number of data points. Compared to the quantile operator, the values previously described from 0.00 to 0.20 are in bin 1, from 0.20 to 0.40 in bin 2, and so on. These discretizers are useful when the feature cannot be described using one single distribution or when observations in one bin behave differently compared to observations in other bins. Numerical features which describe someone's age are good examples of features which are often discretized.

N11: The equal frequency discretizer with 10 bins.

N12: The equal range discretizer with 5 bins. Instead of putting the same number of values in each of the k bins, it is also possible to arrange the bins using an equal range or width for each bin. With $k = 5$, a minimum feature value of 0, and a maximum value of 100, the feature values in the range from 0 to 20 fall into bin 1, from 20 to 40 in bin 2, and so on.

N13: The equal range discretizer with 10 bins.

N14: The bigger-than-zero operator. Many numerical features are count features which also contain a large number of zeroes. Therefore, it can be convenient to split the feature at zero as the feature values which are larger than zero follow a specific distribution which cannot be used properly due to the many zeroes. In order to apply this feature, the feature values in a training set should contain both strictly positive values and values which are less or equal than zero.

$$x_l^* = \begin{cases} 1, & f_{il} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

N15: The median-split operator. Instead of splitting at zero, one can also split the feature in two based on the median of the feature.

$$x_l^* = \begin{cases} 1, & f_{il} > \text{median}(f_i) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Categorical operators. The different categorical operators can also be split. Some operators can be used to transform the categorical feature to a numerical one while others explore different encoding techniques than the standard one-hot encoder. It should be mentioned that categorical features should and have always been one-hot encoded before entering AutoML. It should be mentioned that all categorical features in the raw data sets of OpenML are already label encoded from 0 to k such that the features do not contain text. This means that a sex feature does not contain labels "Male" and "Female" but is already denoted with 0 and 1. Automatic grouping operators have been explored, but were found to be not ideal. The following four categorical operators, C1 to C4, have been implemented for which k is the original number of categories present in a categorical feature:

C1: Cat-to-num operator. Due to the fact that the data in OpenML does not distinguish between ordinal and nominal categorical features, it can help to convert an ordinal feature to a

numerical one besides one-hot encoding the feature. A possible disadvantage of this operator is that it can be pure luck that the categorical values seem ordinal. This is another reason why it can be important in the feature engineering phase to not just use the feature with the highest probability score, as was previously done, but to select between the best predicted new features with a selection method based on the new data set.

C2: Frequency encoder. This operator also transforms the feature to a numerical one. The new values are based on the frequency of each category. If one category has a frequency of 0.23, all the corresponding observations l will get the numerical value 0.23. This is the only categorical encoder for which it can be useful to be applied to a categorical feature with $k = 2$. For all other categorical operators, nothing changes when they are applied to binary features.

C3: Binary encoder. When a feature contains a large number of categories, a one-hot encoder creates many new binary features. This can lead to certain problems. The binary encoder converts each category label to a binary number and creates a column for each binary digit. Consider a feature with 16 categories. The first category label is converted to 0000 and the last one to 1111. This means that for the 16 categories, only four binary features, or in general $\lceil \log_2(k) \rceil$ features, will be created instead of the 15 after one-hot encoding.

C4: Backward difference encoder. There are many smarter encoding techniques called contrast encoders which can work better than one-hot encoding. We have implemented one of these methods which returns $k - 1$ numerical features instead of $k - 1$ binary ones. In backwards difference encoding, the first new column is defined as a certain value for the first category and other values for the other categories. In the second column, the first two categories get the same value and all others another one, and so on. The last column has the same value for the first $k-1$ categories and a different value for the last one. Therefore, it can be interpreted that each column does not look at one category, but at one category compared to the previous one. Contrast encoders work especially good with ordinal data.

Binary operators. The binary operators are written down in the same way as the unary operators. Features f_h and f_i are used with an operator to create the resulting feature set x^*.

Numerical-numerical operators. Most numerical-numerical (NN) operators use basic mathematical operations to create a new feature. As it is possible to also use unary operators as the square root operator within a binary operator, this list can be very extensive. However, an operator as $\sqrt{f_h} - (f_i)^3$ would usually make no sense and we limit ourselves therefore to certain relevant combinations. Furthermore, some operators create a new categorical feature using the two numerical ones. The following nine NN operators have been implemented.

NN1 Plus operator. This operator sums the two features. The average operator does not need to be considered as this is just the plus operator divided by two.

NN2 Minus operator. The order is not important when subtracting the two features as this only changes the sign of the feature and the interpretation of the coefficients, but not the predictive power of the resulting feature.

$$x^* = f_h - f_i \tag{6}$$

NN3 Absolute difference operator. Instead of just subtracting the two features, it is also possible to take the absolute difference.

$$x^* = |f_h - f_i| \tag{7}$$

NN4 Times operator. This operator multiplies the two features.

NN5 Division operator. As mentioned before, we should do protected division (%) instead of normal division. Therefore, it returns the value zero when the denominator is zero. For this operator, the order of the two features is important such that both versions $f_h \% f_i$ and $f_i \% f_h$ should be evaluated. The resulting distribution can mainly differ due to the presence of zeroes in one of the features.

NN6 Hypotenuse operator. This operator calculates the so-called hypotenuse, the long side of a right-angled triangle, using the Pythagorean theorem. It is a combination of the square root, the square, and the plus operator.

$$x^* = \sqrt{f_h^2 + f_i^2} \tag{8}$$

NN7 Ln-times operator. Applying the times operator to two features can easily create a new right-skewed feature when both original features are strictly positive. Therefore, an extra operator is the ln-times operator, which takes the natural logarithm of the multiplication of the two features to solve the possible right-skewness.

$$x^* = \ln(f_h \times f_i) \tag{9}$$

NN8 Larger-than operator. This operator checks which of the two feature values is larger and creates a binary feature using that information. Of course, this operator cannot be applied to two features for which one of them is strictly larger than the other one.

$$x_l^* = \begin{cases} 1, & f_{hl} > f_{il} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

NN9 Quadrant operator. Using the two features, we can also create a categorical feature with four levels depending on the medians of both features. One category in the new feature represents the observations in which both features are large, one in which both features are small, and two category levels represent the case when one feature is large and the other small.

$$x_l^* = \begin{cases} 0, & f_{hl} \leq \text{median}(f_h) \ \& \ f_{il} \leq \text{median}(f_i) \\ 1, & f_{hl} \leq \text{median}(f_h) \ \& \ f_{il} > \text{median}(f_i) \\ 2, & f_{hl} > \text{median}(f_h) \ \& \ f_{il} \leq \text{median}(f_i) \\ 3, & \text{otherwise} \end{cases} \tag{11}$$

Numerical-categorical operators. For the numerical-categorical operators NC1 to NC6, the feature f_h is the numerical one and f_i the categorical one. These operators mostly create a feature based on the different feature values which correspond to the different categories.

NC1 Group-by-then-mean operator. It can be assumed that the observations for different categories differ. Given a categorical and a numerical feature, one can create a feature which contains statistics of the numerical values belonging to different categories. The group-by-then-mean operator calculates the mean of the feature values in f_h for all different categories in f_i. The new feature value for observation l is the mean belonging to category f_{il}.

$$x^* = \text{GroupByThenMean}(f_h, f_i) \tag{12}$$

NC2 Group-by-then-median operator. Instead of the mean of each category, it is also possible that the median or the standard deviation incorporate useful information.

$$x^* = \text{GroupByThenMedian}(f_h, f_i) \tag{13}$$

NC3 Group-by-then-stdev operator. The standard deviation of different groups can give a good indication whether certain categories are more stable than others. Other statistics of the numerical feature, such as the minimum value, have been evaluated by other researches but were found to be not very useful.

$$x^* = \text{GroupByThenStdev}(f_h, f_i) \tag{14}$$

NC4 Minus-group-median operator. The created feature of operator NC2 can be used again with the numerical feature in order to create a new feature. For each observation, the new feature is defined by subtracting the median in the category to which the observation belongs from the numerical feature.

$$x^* = f_h - \text{GroupByThenMedian}(f_h, f_i) \tag{15}$$

NC5 Group-standardize operator. The mean and standard deviation of the numerical feature values belonging to each category can be used to standardize the feature differently for each category.

$$x^* = (f_h - \text{GroupByThenMean}(f_h, f_i))/\text{GroupByThenStdev}(f_h, f_i) \tag{16}$$

NC6 One-hot-numerical operator. This last operator splits the numerical feature into k columns based on f_i. A column referring to a certain category contains the numerical value if that observation belongs to that category and zero otherwise.

$$x^* = f_h \times \text{OneHotEncoder}(f_i) \tag{17}$$

Categorical-categorical operators. This subset of binary operators will not be considered in this paper as hardly any categorical-categorical operator has been considered in recent literature and it can be said that most ML algorithms can use interactions between two categorical features by themselves.

2 Meta-Features

In order to create a good meta-model, there should also be a complete meta-data set with a set of relevant meta-features. All the different data sets and their features are of various sizes such that the different features should be generalized to usable meta-features using several feature characteristics. The information that some features which follow a certain distribution result in a positive training sample for an operator while others with a different distribution do not, can be used when training the meta-model for that operator. In the next paragraphs, we will name all meta-features which will in total form the meta-data for the different types of operators. All formulas can be found afterward in Table 1.

Numerical Meta-features. The first four meta-features for a numerical feature are its first four moments: the mean, the variance or standard deviation, the skewness, and the kurtosis. Besides these characteristics, the median, minimum, maximum, range size, mode, and mode frequency of a feature are also saved as meta-features. Another meta-feature denotes whether the original feature contains both positive and negative values or not.

Just like the only characteristic saved in LFE, we also use an equal range discretizer to convert the feature to a number of meta-features denoting the feature frequency in those bins. For the meta-data corresponding to numerical operators, the number of bins is chosen to be 10. This means that a perfectly uniform distributed feature gets ten meta-features of 0.1 whereas a normally distributed feature has higher values in meta-features 5 and 6 of the 10 values in total. These bins are also used with a local-maxima-finding algorithm to find the number of distinct peaks in this list, given that the peak should be higher than 0.1. This meta-feature can be used to discover feature distributions which do not follow a standard distribution as they have multiple peaks and can, therefore, indicate that the feature should be discretized.

The next group of characteristics is based on the interaction between the feature and the target variable of the data set. The first meta-feature is the correlation between the two features. Moreover, instead of creating a list with the binned frequencies of the whole feature, two lists can be created after the feature is split in two using the target variable as was also done with QSA. One list with the ten binned frequencies for the values belonging to $y = 1$ and one belonging to $y = 0$. The bin ranges are of course decided using the complete feature.

These two sets of values can produce different meta-features which can give a good indication of whether the data points belonging to one class are different than the ones belonging to the other class. One of them is the correlation between both sets while another one denotes the absolute difference between the bin identifiers with the maximum frequency within the sets. If for one class the densest bin is bin number 5 and for the other class it is bin number 3, this meta-feature returns a value of 2, stating that there is a hint that the distributions differ. A supporting measure is the rate of means for all numerical values in the two classes, which is always calculated as the highest mean divided by the other mean. The final characteristic using these sets of binned frequencies is the chi-square two sample test statistic, which is a test statistic for binned data in order to see if two data samples come from the same distribution.

In total, this means that any numerical feature is transformed into one meta-target variable and 27 meta-features. An overview of the set of meta-features for a numerical feature can be found in Figure 1.

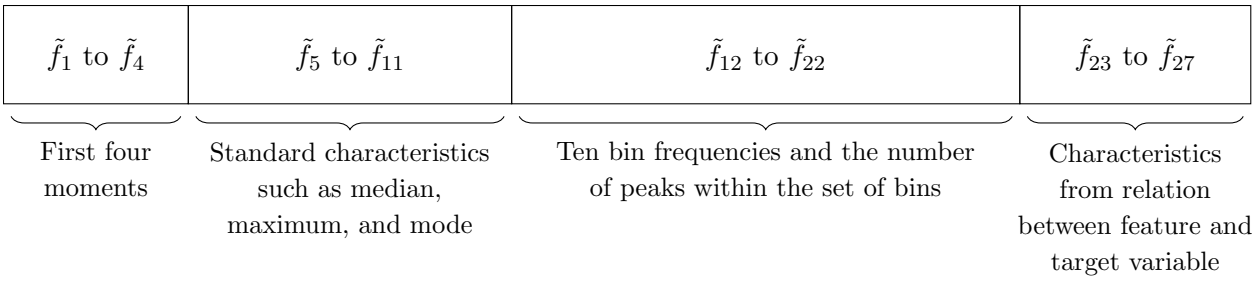

Figure 1: The different meta-features created for every original numerical feature.

Categorical Meta-features. For a categorical feature, several different characteristics are saved as meta-features. The first meta-feature is the number of categories. Next, we will look at the frequencies of the different categories. The frequencies of the four most occurring categories and the total frequency of the remaining categories are saved. These values can be zero when the number of categories is low. Two other meta-features denote the minimum frequency and the maximum frequency divided by the minimum frequency.

Again, information about the target variable can be used to create certain meta-features. The correlation between the target variable and the categorical feature can give a good indication if the categorical feature is ordinal or not. Moreover, the feature can again be split in two based on the two classes and represented using the frequencies within the two sets for the top occurring categories in the whole feature. In this representation, it can easily be seen whether certain categories contain relatively more ones or zeroes in the target variable than it is the case in other categories. The saved characteristics are the correlation between the two sets of frequencies, an indication which shows if the maximum frequency belongs to the same category, and the maximum difference between frequencies for the same category.

Consider a categorical feature with four categories and sorted frequencies for these categories 0.5, 0.25, 0.15, and 0.1. When split based on the target classes, the two sets of frequencies can look like this. $y = 1$: (0.7, 0.15 0.1, 0.05) and $y = 0$: (0.3, 0.35, 0.2, 0.15). The first and largest category mostly correspond to the target values of 1 such that for $y = 0$, the second category is the largest category within that subset. This means that the last two meta-features mentioned become 0 and 0.28 because the maximum frequency does not belong to the same category and because 0.28 is the maximum difference between two category frequency levels $(0.6 - 0.32)$. In total, 12 meta-features belonging to a categorical feature are being saved and an overview of these meta-features is given in Figure 2.

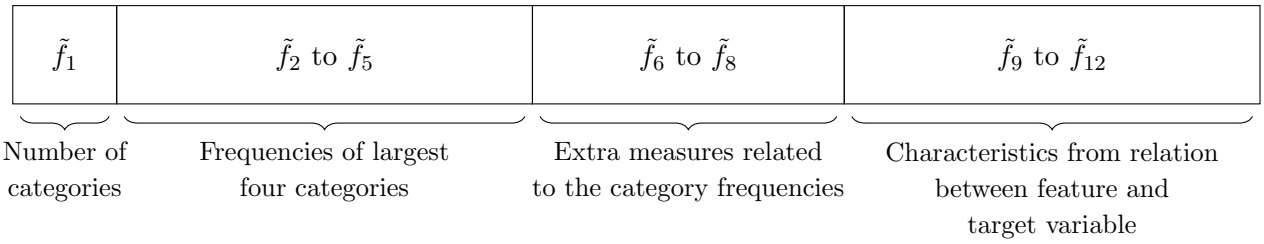

Figure 2: The different meta-features created for every original categorical feature.

Numerical-numerical Meta-features. The 41 meta-features which are created for the meta-data of numerical-numerical operators mainly consists of characteristics from the features f_h and f_i. A meta-observation firstly consists of characteristics of feature h, then ones of feature i, and finally some characteristics of features h and i together. This is visualized in Figure 3.

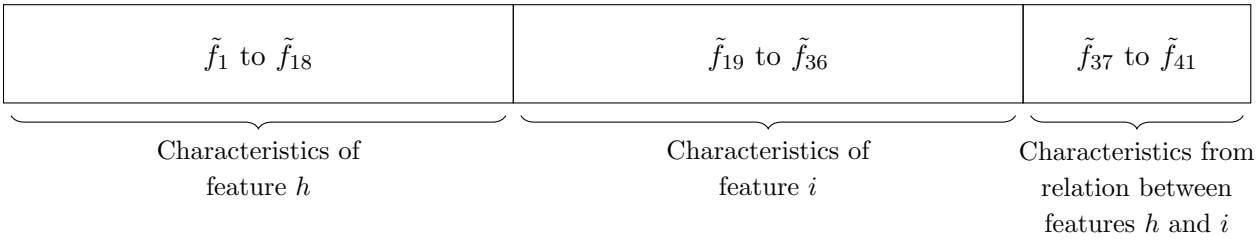

Figure 3: The different meta-features created for every pair of numerical features.

It can be seen that within this observation, the order of h and i matters. In extreme cases, the prediction probability of the effectiveness of an operator on a pair of features can be influenced by this order. For example, when the meta-model is trained such that it returns a 1 when f_h is normally distributed and f_i is exponentially distributed, it can cause a problem when f_h follows an exponential distribution and f_i a normal one. This immediately brings us to solving the problem addressed earlier about certain operators in which the order should be taken into account. The solution is to always save two meta-observations for each pair of numerical features and to always evaluate both orders for any new numerical feature pair.

For numerical-numerical operators, a Boolean variable is saved denoting whether the operator should be processed in both ways. If this is not the case, the meta-target variable $\tilde{y}$ is definitely the same for both orders. Both meta-observations [$\tilde{y}$; $\tilde{f}_1$ to $\tilde{f}_{18}$; $\tilde{f}_{19}$ to $\tilde{f}_{36}$; $\tilde{f}_{37}$ to $\tilde{f}_{41}$] and [$\tilde{y}$; $\tilde{f}_{19}$ to $\tilde{f}_{36}$; $\tilde{f}_1$ to $\tilde{f}_{18}$; $\tilde{f}_{37}$ to $\tilde{f}_{41}$] are added to the meta-data. For operators as the division operator which should be processed in both ways, the same is done but with a possibly different meta-target variable $\tilde{y}$ in the two meta-observations. In total, this means that for each pair of numerical features, two meta-observations are always made. Later, during the feature engineering phase, both orders should be evaluated for each new pair.

For a numerical feature, fewer characteristics, only 18 instead of 27, are saved compared to the ones described in the section about the numerical meta-features. Compared to the meta-features described in that section, the following nine characteristics are not saved. The largest impact is that five bins are saved instead of ten bins. Furthermore, the meta-features denoting the number of peaks in these bins, the kurtosis, whether the feature contains both positive and negative values, and the difference between the bins with the maximum frequency in the two sets after splitting the data using the target variable are not considered in order to reduce the dimension of the final meta-observation to make the last set of characteristics more important.

This last set of characteristics are several interactions between features f_h and f_i. These meta-features are the correlation between the two features, the rates of the means, standard deviations, and ranges and a meta-feature denoting if the largest frequency in the five bins for both features belongs to the same bin or not. Using this information, it is possible to get an indication if both features follow a similar distribution or not.

Numerical-categorical Meta-features. The numerical-categorical meta-data does not need to be saved twice as f_h is always the numerical one and f_i the categorical one. For a numerical feature, the same characteristics are saved as the ones described for a numerical feature in the numerical-numerical meta-data sets. For the categorical feature, the same information is saved as the information described in the section about the categorical meta-features.

Finally, two meta-features are created about the interaction between f_h and f_i. The correlation between both features is saved as well as a measure about the different means of f_h which belong to the most frequent categories in f_i. Consider a categorical feature with k values. The numerical feature values belonging to the top three categories of that feature have, for example, means 85, 70, and 75. The meta-feature which is saved denotes the maximum mean within the top three categories divided by the minimum mean, which is in this case 85 divided by 70. This meta-feature can give an indication whether the numerical values differ for different categories such that an operator can be more meaningful. The numerical-categorical meta-data sets will in total contain 32 meta-features and a visualization can be found in Figure 4.

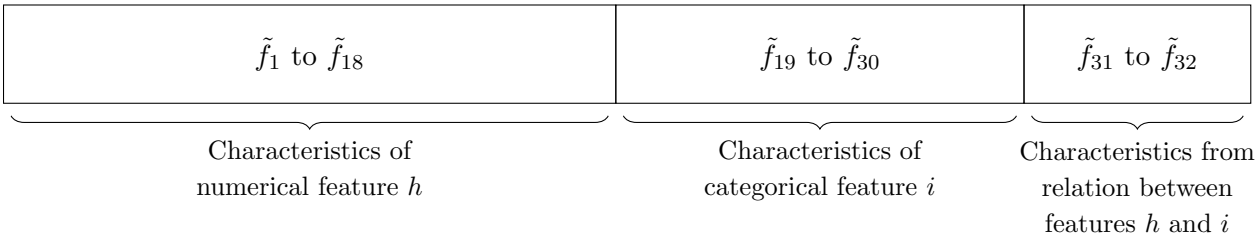

Figure 4: The different meta-features created for every pair of a numerical and a categorical feature.

In Table 1, we present all meta-features used in this research. In the first column, the formula of the meta-feature is given and in the last columns it is shown in which meta-observation that meta-feature is used. In these formulas, f_N is a numerical feature, f_C is a categorical feature, f_{N2} is the second numerical feature, necessary in numerical-numerical meta-observations, and y is the original target variable of the data set. For the numerical-numerical meta-observations, 2×1 means that the formula is applied to both numerical features.

Within the formulas, BIN(f_N) means that this formula produces the binned frequencies after the numerical feature is binned using equal ranges. The number of bins, which can be ten or five, is given in the other columns with numbers larger than one. BIN($f_N|y = 1$) and BIN($f_N|y = 0$) are the binned frequencies for the observations with $y = 1$ and $y = 0$ respectively. For the categorical variables, BIN_top4(f_C) or BIN_top5(f_C) means that the frequencies of the top 4 or top 5 categories are returned.

Table 1: All meta-features used in the meta-data for the different operators. N = Numerical, C = Categorical, NN = Numerical-numerical, and NC = Numerical-categorical.

Formula of the meta-feature	N	C	NN	NC
mean(f_N)	1		2×1	1
standard_deviation(f_N) = stdev(f_N)	1		2×1	1
skewness(f_N)	1		2×1	1
kurtosis(f_N)	1			
median(f_N)	1		2×1	1

$\min(f_N)$	1		2×1	1
$\max(f_N)$	1		2×1	1
$\text{range}(f_N) = \max(f_N) - \min(f_N)$	1		2×1	1
$\text{mode}(f_N)$	1		2×1	1
$\text{freq}(\text{mode}(f_N))$	1		2×1	1
$\begin{cases} 1, & \text{if } \min(f_N) < 0 \ \& \ \max(f_N) > 0 \\ 0, & \text{otherwise} \end{cases}$	1			
$\text{BIN}(f_N)$	10		2×5	5
$\text{number\_of\_peaks}(\text{BIN}(f_N))$	1			
$\text{corr}(f_N, y)$	1		2×1	1
$\text{corr}(\text{BIN}(f_N \mid y = 1), \text{BIN}(f_N \mid y = 0))$	1		2×1	1
$\lvert \text{maxbin}(\text{BIN}(f_N \mid y = 1)) - \text{maxbin}(\text{BIN}(f_N \mid y = 0)) \rvert$	1			
$\text{chisquare\_testscore}(\text{BIN}(f_N \mid y = 1), \text{BIN}(f_N \mid y = 0))$	1		2×1	1
$\begin{cases} \text{mean}(f_N \mid y = 1)/\text{mean}(f_N \mid y = 0), & \text{if mean}(f_N \mid y = 1) \geq \text{mean}(f_N \mid y = 0) \\ \text{mean}(f_N \mid y = 0)/\text{mean}(f_N \mid y = 1), & \text{otherwise} \end{cases}$	1		2×1	1
$\text{number\_of\_categories}(f_C)$		1		1
$\text{BIN\_top4}(f_C)$		4		4
Sum of remaining categories = 1 - $\text{sum}(\text{BIN\_top4}(f_C))$		1		1
$\text{freq}(\text{smallest\_category}(f_C))$		1		1
$\text{freq}(\text{largest\_category}(f_C)) \ / \ \text{freq}(\text{smallest\_category}(f_C))$		1		1
$\text{corr}(f_C, y)$		1		1
$\text{corr}(\text{BIN\_top5}(f_C \mid y = 1), \text{BIN\_top5}(f_C \mid y = 0))$		1		1
$\lvert \text{maxbin}(\text{BIN\_top5}(f_C \mid y = 1)) - \text{maxbin}(\text{BIN\_top5}(f_C \mid y = 0)) \rvert$		1		1
$\max(\lvert \text{BIN\_top5}(f_C \mid y = 1) - \text{BIN\_top5}(f_C \mid y = 0) \rvert)$		1		1
$\text{corr}(f_N, f_{N2})$			1	
$\begin{cases} \text{mean}(f_N)/\text{mean}(f_{N2}), & \text{if mean}(f_N) \geq \text{mean}(f_{N2}) \\ \text{mean}(f_{N2})/\text{mean}(f_N), & \text{otherwise} \end{cases}$			1	
$\begin{cases} \text{stdev}(f_N)/\text{stdev}(f_{N2}), & \text{if mean}(f_N) \geq \text{mean}(f_{N2}) \\ \text{stdev}(f_{N2})/\text{stdev}(f_N), & \text{otherwise} \end{cases}$			1	
$\begin{cases} \text{range}(f_N)/\text{range}(f_{N2}), & \text{if mean}(f_N) \geq \text{mean}(f_{N2}) \\ \text{range}(f_{N2})/\text{range}(f_N), & \text{otherwise} \end{cases}$			1	
$\begin{cases} 1, & \text{if maxbin}(\text{BIN}(f_N)) = \text{maxbin}(\text{BIN}(f_{N2})) \\ 0, & \text{otherwise} \end{cases}$			1	
$\text{corr}(f_N, f_C)$				1
$\max(q) \ / \ \min(q) \mid q = \text{GroupByThenMean}(f_N, \text{largest\_four\_categories}(f_C))$				1

3 Meta-Models

Based on the hyperparameter options used in AutoML for a random forest classifier and the ranges and options found in case studies online, the following parameter grid was chosen to tune the random forest. In Table 2, all hyperparameters and their possibilities which were explored for creating the meta-models are shown. The random forest classifier has even more hyperparameters, but those are usually not considered when tuning this classifier. The values of the hyperparameter `n_estimators` were chosen to be quite high and similar for comparison reasons. This hyperparameter denotes the number of trees in a forest. It is unjustifiable to compare a probability from a model which uses ten trees with a model which uses a couple of hundred trees due to the error margins of the resulting probabilities.

Table 2: All hyperparameters and their values which are tuned within the random forest classifiers in order to create the meta-models.

Hyperparameter	Possibilities
`bootstrap`	{True, False}
`criterion`	{gini, entropy}
`max_depth`	{3, 5, 10, None}
`max_features`	{1, 5, 10, sqrt, 0.5}
`min_samples_leaf`	{1, 3, 5}
`min_samples_split`	{2, 4, 7, 10}
`n_estimators`	{199, 200, 201}