Machine Learning

Exercise 1: Classification

Emile Johnston 12229987

Fani Sentinella-Jerbić 12206591

Daria Stefan 12229258

April 2023

Contents

1 Approach

1.1 Datasets

1.1.1 Freely chosen datasets

Here we chose two very different datasets. The first one deals with predicting the survival of hepatitis patients. It offers an interesting perspective on using medical data to potentially improve decision-making in the health sector. With a relatively small number of instances, it makes a good contrast with the second dataset we chose, for which the task is to sort a given bean into one of the seven target classes, according to its shape. The shape is given by numerical data generated from photos of beans. We chose this dataset mainly because it is more complex than the hepatitis dataset, offering us the chance to expand our skill sets. Both datasets require scaling of attributes. The first dataset also requires dealing with missing values.

Hepatitis dataset:

Instances: 155	**Task**: binary prediction
Attributes: 19	**Required preprocessing**: missing values & scaling

Table 1: Attribute description for the Hepatitis dataset

attribute	data type	possible values / value ranges	number of missing values
AGE	ratio	7 – 78	0
SEX	nominal	male, female	0
STEROID	nominal	no, yes	1
ANTIVIRALS	nominal	no, yes	0
FATIGUE	nominal	no, yes	1
MALAISE	nominal	no, yes	1
ANOREXIA	nominal	no, yes	1
LIVER BIG	nominal	no, yes	10
LIVER FIRM	nominal	no, yes	11
SPLEEN PALPABLE	nominal	no, yes	5
SPIDERS	nominal	no, yes	5
ASCITES	nominal	no, yes	5
VARICES	nominal	no, yes	5
BILIRUBIN	ratio	0.3 – 8	6
ALK PHOSPHATE	ratio	26 – 295	29
SGOT	ratio	14 – 648	4
ALBUMIN	ratio	2.1 – 6.4	16
PROTIME	ratio	0 – 100	67
HISTOLOGY	nominal	no, yes	0
Class	nominal	DIE, LIVE	0

Dry beans dataset:

Instances: 13611 **Task**: multiclass classification

Attributes: 17, all ratio **Required preprocessing**: scaling

Table 2: Attribute description for the Dry Bean dataset

attribute	value range	definition
Area (A)	20420 – 254616	The area of a bean zone and the number of pixels within its boundaries
Perimeter (P)	524.736 – 1985.37	Bean circumference is defined as the length of its border
MajorAxisLength (L)	183.6011650038393 – 738.8601534818813	The distance between the ends of the longest line that can be drawn from a bean
MinorAxisLength (l)	122.51265345074418 – 460.1984968278401	The length of the longest line that can be drawn from the bean while standing perpendicular to the main axis
AspectRatio (K)	1.0248675960667681 – 2.430306446836626	$K = \frac{L}{l}$
Eccentricity (Ec)	0.21895126335356507 – 0.9114229684680053	Eccentricity of the ellipse having the same moments as the region.
ConvexArea (C)	20684 – 263261	Number of pixels in the smallest convex polygon that can contain the area of a bean seed.
EquivDiameter (Ed)	161.24376423134018 – 569.3743583287609	The ratio of the pixels in the bounding box to the bean area $Ed = \sqrt{\frac{4A}{\pi}}$
Extent (Ex)	0.55531471681117 – 0.8661946405648266	The ratio of the pixels in the bounding box to the bean area
Solidity (S)	0.9192461570857022 – 0.9946774999456888	$S = \frac{A}{C}$
Roundness (R)	0.4896182562412148 – 0.9906853996160323	$R = \frac{4A\pi}{p^2}$
Compactness (CO)	0.6405767589768725 – 0.9873029693778109	$Co = \frac{Ed}{L}$
ShapeFactor1	0.0027780126683855494 – 0.010451169324378654	$\frac{L}{A}$
ShapeFactor2	0.0005641690180332927 – 0.0036649719644516834	$\frac{l}{A}$
ShapeFactor3	0.41033858414131424 – 0.9747671533422431	$\frac{A}{\pi \frac{L}{2}^2}$
ShapeFactor4	0.9476874027098624 – 0.9997325300471389	$\frac{A}{\pi \frac{Ll}{4}}$
Class	Seker, Barbunya, Bombay, Cali, Dermosan, Horoz, Sira	target classes are nominal

1.1.2 Kaggle datasets

Again, we have two datasets with very different sizes: the breast cancer dataset has 32 columns and 285 rows, while the loan dataset has 78 columns and 10 000 rows.

Breast Cancer. The dataset contains **30 numerical attributes**, which correspond to the mean, standard error, and worst (largest) values of 10 different characteristics of the breast tissue cells, namely radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. Other than the numerical attributes, the dataset additionally contains an identificator attribute and the **class** attribute indicating whether the breast tissue is **malignant** ("true") or **benign** ("false"). The class distribution is **imbalanced** with 177 benign and 108 malignant observations.

Loan. The dataset represents loan applications and their characteristics, such as the loan amount, term, interest rate, etc, and information about the applicants, like home ownership. There are ID numbers, 12 categorical variables (including the target class "grade"), 4 interval variables, 1 ordinal variable and 74 ratio variables. The distribution of the target class is very imbalanced, with the number of occurrences going from 57 for grade G to 2989 for grade C.

1.2 Preprocessing Methods

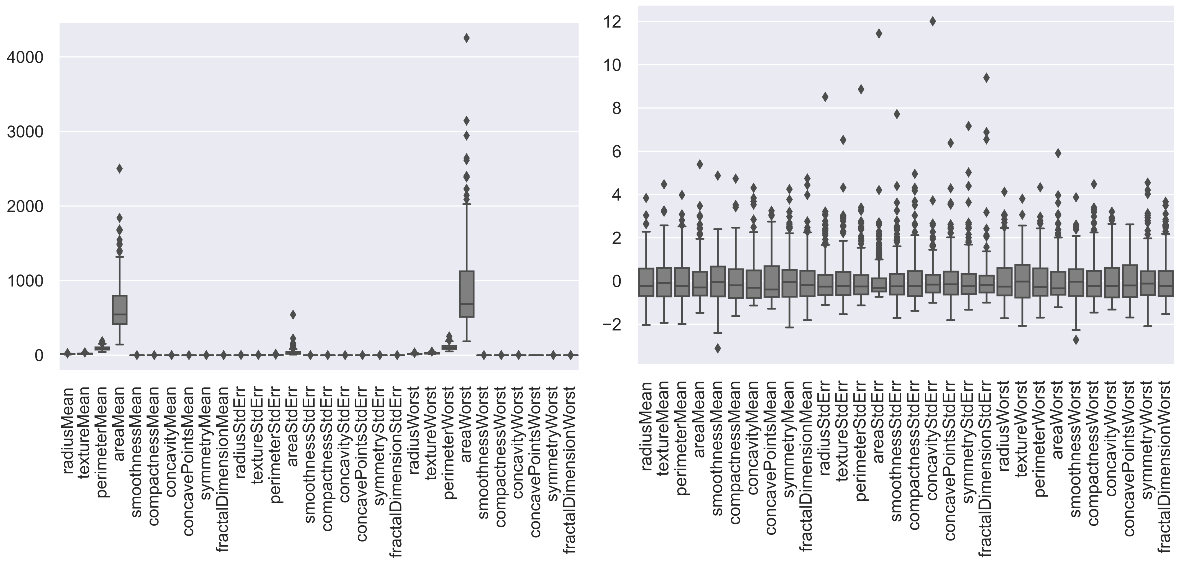
Exploration. For all datasets, we begin by taking a look at the raw data to see if it has any particularities and if we can work with it properly. We also calculated some descriptive statistics such as mean, median and standard deviation to get familiar with the numbers.

Missing values. We explicitly check every cell to see if it has the kind of value it should. For numerical attributes, we iterate through each row and each column and see if the cell has the same type as the rest of the column, integer or float. For categorical attributes, missing values could potentially be encoded as data of type object so this strategy can not be applied. Instead, we manually checked the contents of the columns. This set had no missing or poorly formatted values.

Data imputation. When there are missing values, we proceed with data imputation. Since we can't know in advance which imputation method is the best, we try several of them and see what influence they have on the results.

Categorical to numerical. Certain algorithms like SVM and logistic regression need to work on numbers only, so when there are categorical attributes, we need to make them numerical. When they don't correspond to equidistant values, the best solution is one-hot encoding. For a categorical attribute with n distinct values, we replace the column with n columns that are filled with 1 for `true` and 0 for `false`.

Scaling. The different attributes of each dataset are often on very different scales. This can be a problem for algorithms which rely on the distance between data points, such as SVM (Support Vector Machine) and logistic regression. So we will observe the effects of scaling by comparing no scaling to min-max scaling and standard scaling for each dataset. We do this by including a `scale` parameter in our functions, which is associated to the used model to form a `pipeline` object which will automatically scale the data when it is being processed by the model. Min-max and standard scaling are the most common scaling methods and are suitable for the classifiers we're using: with min-max we make all attributes have the same minimum and maximum values, while with standard scaling we make them all have 0 as a mean and 1 as standard deviation. Below is an illustration of what standard scaling does to one of our datasets:



(a) Before standard scaling

(b) After standard scaling

Figure 1: Boxplots of the Breast Cancer numerical attributes

1.3 Classifiers

Choice of classifiers. We needed to choose three classifiers, so we chose among the most well-known ones. What mattered with the classifiers is that they perform well on the datasets for which we are in a competition, so the breast cancer dataset and the loan dataset. To get a feel for this, we experimented with a few different classifiers, with default parameters. We found that for the breast cancer dataset, SVM seemed like the most effective classifier, and for the loan dataset it was the decision tree. So we kept these two, and also logistic regression, because it's quite different from the other algorithms, but performs well. SVM and logistic regression require data that is all on the same scale, and that is all numeric, to work properly, while the decision tree requires neither of those.

1.4 Model Evaluation and Selection

Choice of metrics. We have two datasets on which we must perform binary classification and two datasets on which we must perform multi-class classification (7 classes). In the binary case, the usual metrics are accuracy, recall, precision, f1-score and f2-score. In the multi-class case, the usual metrics are accuracy, micro f1-score, and macro f1-score. We calculate some of these, based on the specificity of each dataset, which we explain further on.

Holdout vs. cross-validation. We want to compare different methods for the evaluation of models, namely the holdout method and the cross-validation method. To do this, we made one function for each method, where the function trains a given model with a given scaling method, computes the results, and returns the relevant metrics. We made sure that everything is the same in both functions, except the splitting method, so that our comparison can be reliable. We also made sure to try the hold-out method multiple times with different random seeds to give it some variability and robustness as well as to make it more comparable to cross-validation.

Choice and analysis of hyperparameters. For each algorithm we considered a set of hyperparameters that are commonly used. Using the technique of grid search, we evaluated the performance of the models on all combinations of these hyperparameters with cross-validation. To illustrate the effects of each hyperparameter on the performance of the model, we took an agnostic approach. We wanted to be able to make conclusions about all hyperparameters using only one visualization. For this purpose, we used the **parallel coordinates plot** where the coordinates show different hyperparameter values and the color shows the performance of the model. This way it is easy to see how a certain value of a hyperparameter affects the performance of the model. The visualization is even interactive which can be useful for choosing a specific range of a value and inspecting it further, but this, of course, we are not able to illustrate in this static report.

Note. We even considered and implemented nested cross-validation with model selection. However, this approach took too much time even for the small datasets, so we settled on doing a cross-validation grid search on training data. We do note that this means the scores gotten using our approach are not a completely fair estimate of the generalization power of models, but we believe it is good enough considering the purpose and computational abilities.

2 Experimentation

2.1 Freely Chosen datasets

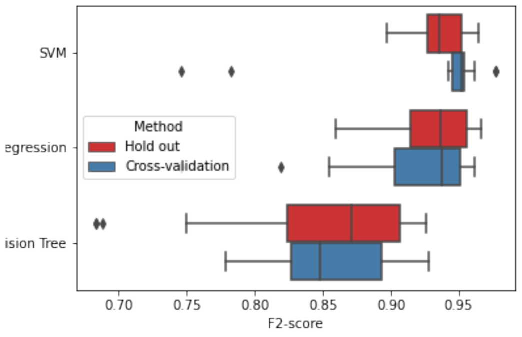
2.1.1 Hepatitis

Preprocessing. First, we take a quick look at the raw data to get familiar with the dataset and check for particularities. We notice quite a few missing values, marked with a question mark, but these don't seem to be a majority in any column. We decided to tackle this issue with imputation using the mean value and deleting the columns containing all missing values. However, since many columns represent binary variables imputation of the mean wouldn't make much sense for these. For this reason, we decided to additionally round up the values of imputed means to either 0 or 1. As for scaling, we will compare the performance of our classifiers on three sets of features: the original one, one scaled with the StandardScaler, and the one scaled with the MinMaxScaler.

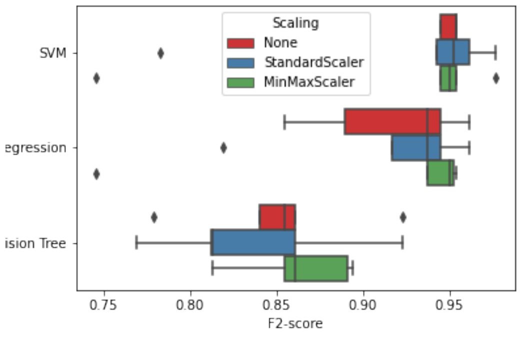
Performance measures. As we are dealing with medical data, which could determine life or death in some cases, we prefer to have more patients diagnosed with cancer as opposed to missing out on patients with cancer. In other words, we would prefer more false positives than false negatives. Because of this, we will be using **F2 score** as our main measure here. Unlike the F1 score, which gives equal weight to precision and recall, the F2 score gives more weight to recall than to precision. This is exactly what we need in this case. Additionally, we will be tracking the accuracy, F1 score, and recall for reference and the ability to possibly draw conclusions across different datasets in the end. Lastly, we will be tracking the training duration to see which algorithms are the most time-efficient.

Experimental results. First, we compare the algorithms with default hyperparameters using holdout and cross-validation. The result is shown in Figure 2a. With default settings Logistic Regression and SVM perform well with regards to F2-score, whereas Decision Tree seems to be lagging behind the two. SVM also shows the least deviation from the median. Next, we compare scaling methods as seen in Figure 2b. For SVM, we see the biggest improvement when using standard scaling. Logistic Regression and Decision Tree show the most improvement when using min-max scaling. Lastly, we perform hyperparameter tuning. The results of the grid search for each algorithm are shown in Figure

3, and the final F2-scores of the best models per each algorithm are given in Table 3. SVM seems to be significantly better than the other two.

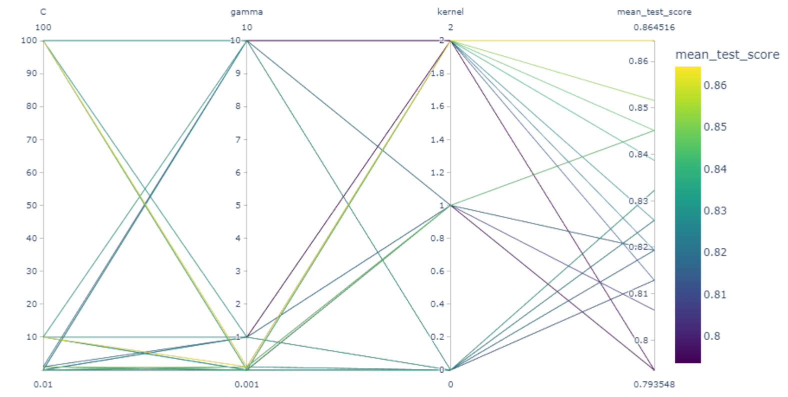


(a) Comparison of evaluation methods

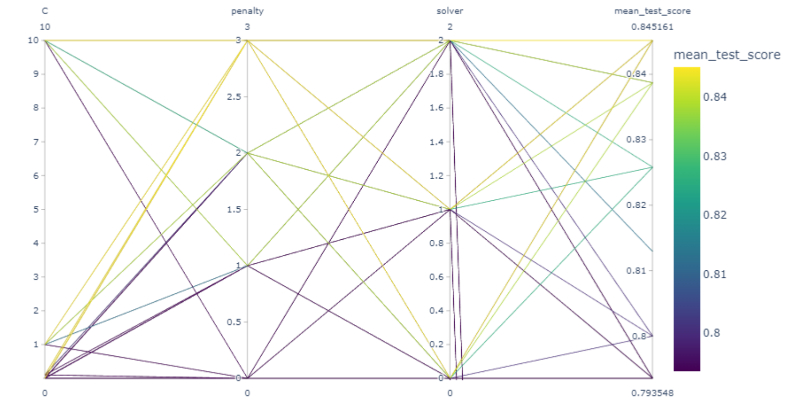


(b) Comparison of scaling methods

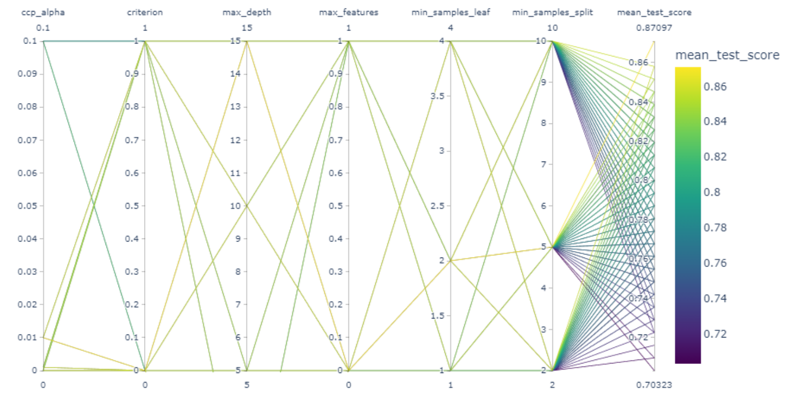
Figure 2: Comparison of different methods on the Hepatitis dataset



(a) **SVM.** Categorical legend –kernel: 0 → linear, 1 → poly, 2 → rbf



(b) **Logistic Regression.** Categorical legend – penalty: 0 → elasticnet, 1 → l1, 2 → l2, 3 → none; solver: 0 → lbfgs, 1 → sag, 2 → saga



(c) **Decision Tree.** Categorical legend – criterion: 0 → entropy, 1 → gini; max features: 0 → log2, 1 → sqrt

Figure 3: Hyperparameter analysis for Hepatitis

	F2-score			
	min	mean	max	std
Algorithm				
Decision Tree	0.708333	0.817792	0.944882	0.109994
Logistic Regression	0.782609	0.908639	0.961538	0.072272
SVM	0.847458	**0.941210**	0.976563	0.053966

Table 3: Final comparison with best scaling scheme and tuned hyperparameters for the Hepatitis dataset

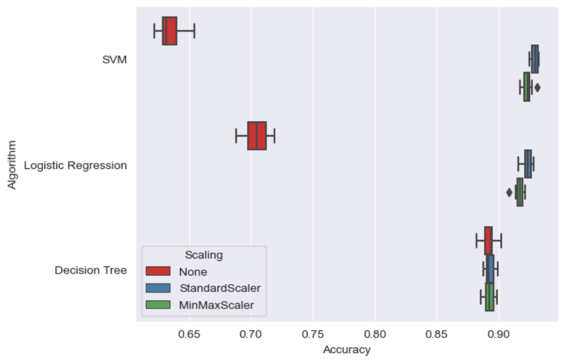
2.1.2 Dry Beans

Preprocessing. Again, we look at the raw data, and nothing seems abnormal. Here it's simple to check for missing values, since all 15 columns (not counting the class) are numerical. Columns 0 and 6 (counting the python way, so the first and seventh column) are of type integer, while all the others are of type float. We found no missing values.

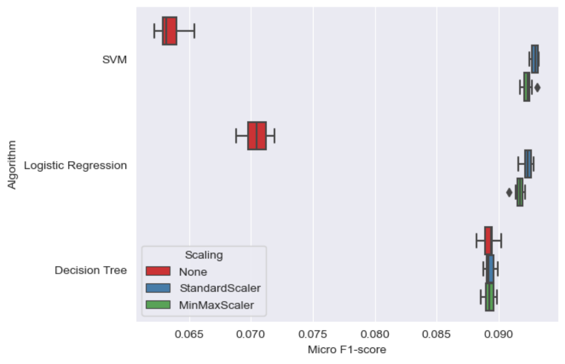
One quick look at the raw data is enough to see that the scales of the different attributes are vastly different, and that our scaling function will come in useful.

Performance measures. Here the goal is to correctly identify the species of a bean. We have no reason to believe that wrongly classifying species X is worse than wrongly classifying species Y, or that wrongly classifying species X as species Y is worse than wrongly classifying species X as species Z. So we can assume that there is no further measure of performance than the fact that correct classification is good and, incorrect is bad. For this purpose, accuracy is the most relevant metric, as it just gives us the proportion of correct predictions. So accuracy will be our main metric for this dataset, but for the sake of completeness we will also compute micro f1-score and macro f1-score, because these try to balance out the score between classes, which might give slightly different results given that the target class distribution is imbalanced.

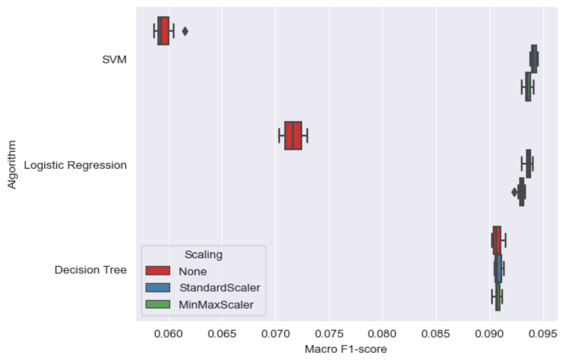
Experimental results. We start with the holdout method, and we train our three models on a portion of the dataset with our three different scaling methods. Since we perform this holdout operation ten times, we calculate the average accuracy, micro f1-score, macro f1-score and training time over the ten splits, and we show the results below with boxplots, to give an idea not only of the average performance but also of the distributions:



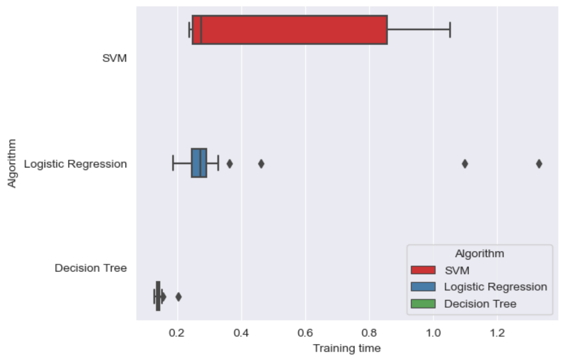
(a) Comparison of accuracy



(b) Comparison of micro f1-score



(c) Comparison of macro f1-score

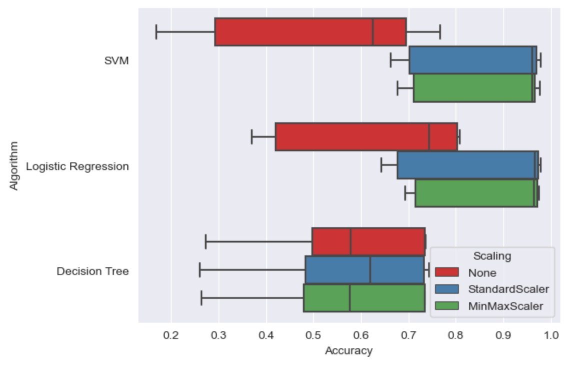


(d) Comparison of training time

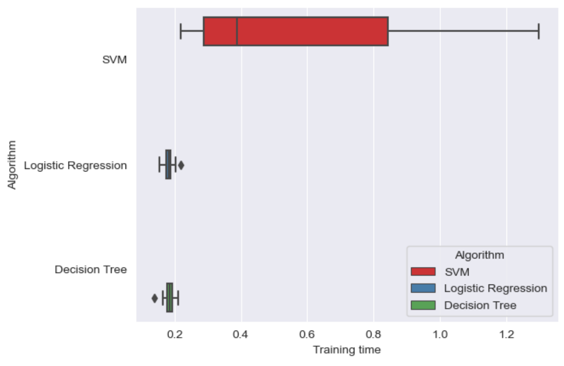
Figure 4: Comparison of different scaling methods and different algorithm for four metrics on the Dry Beans dataset

First of all we can notice that the accuracy, micro f1-score and macro f1-score produce extremely similar results, so the micro and macro f1-scores don't bring us much new information, and the accuracy alone is sufficient to evaluate the models. For SVM and logistic regression, standard scaling is clearly better than min-max scaling, and both are far better than no scaling. For decision tree on the other hand, there is no significant difference between the three methods, but min-max scaling seems very slightly better. Overall, SVM is the algorithm that performs the best, when we use the best scaling method for each algorithm.

Next, we do the same with the cross-validation method:



(a) Comparison of accuracy



(b) Comparison of training time

Figure 5: Comparison of different scaling methods and different algorithms for four metrics on the Dry Beans dataset

Here there is a lot more overlap in the results, but we still get the same conclusions. Then, we compare the two methods:

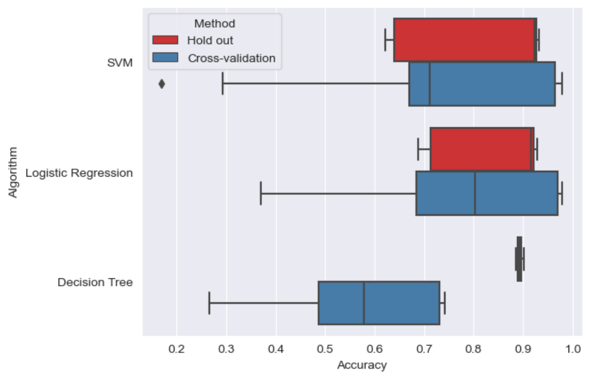


Figure 6: Comparison of accuracy

Figure 7: Comparison of holdout and cross-validation, and different algorithms on the Dry Beans dataset

The holdout method performs clearly better, especially for the decision tree. For SVM and logistic regression, there is a big overlap between the two methods, and holdout seems better, but not significantly.

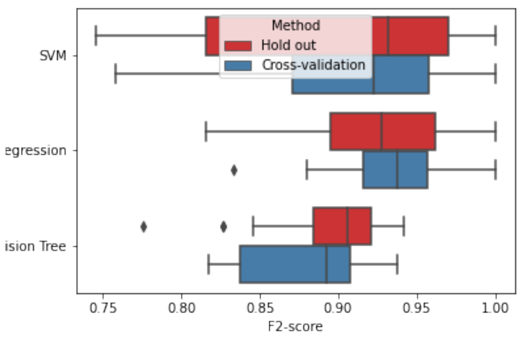
2.2 Kaggle Datasets

2.2.1 Breast Cancer

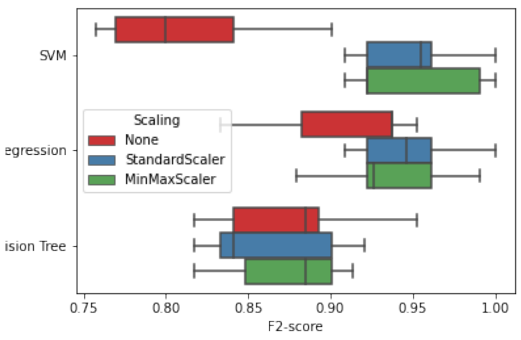
Preprocessing. The dataset doesn't contain any missing values. However, the range of the values differs across the attributes. The extent of this difference is large, with attributes related to the area and the perimeter having significantly larger values and value ranges compared to the rest of the attributes. As previously mentioned in Section 1, our chosen method for mediating these differences is trying min-max and standard scaling schemes.

Performance measures. Same as in the case of the Hepatitis dataset, we are dealing with medical data, and we prefer to have more patients diagnosed with cancer as opposed to missing out on patients with cancer. Because of this, we will again be using **F2 score** as our main measure here. Additionally, we will be tracking the accuracy, F1 score, and recall. Lastly, we will be tracking the training duration to see which algorithms are the most time-efficient.

Experimental results. First, we compare the algorithms with default hyperparameters using hold-out and cross-validation. The result of this comparison is shown in Figure 8a. With default settings Logistic Regression and SVM perform well with regards to F2-score, whereas Decision Tree seems to be lagging behind the two. However, SVM shows the biggest deviation from the median value which could mean it is more unstable than the other two. In most cases, hold-out seems to be more optimistic so in the rest of the experiments we prefer cross-validation. Next, we compare scaling methods as seen in Figure 8b. For SVM, we see the biggest improvement when using min-max scaling. Logistic Regression shows the most improvement when using standardization, and Decision Tree shows doesn't show improvement but later when using hyperparameter tuning, it achieves the best score when combined with min-max scaling. Lastly, we perform hyperparameter tuning. The results of the grid search for each algorithm are shown in Figure 9, and the final F2-scores of best models per each algorithm are given in Table 4.

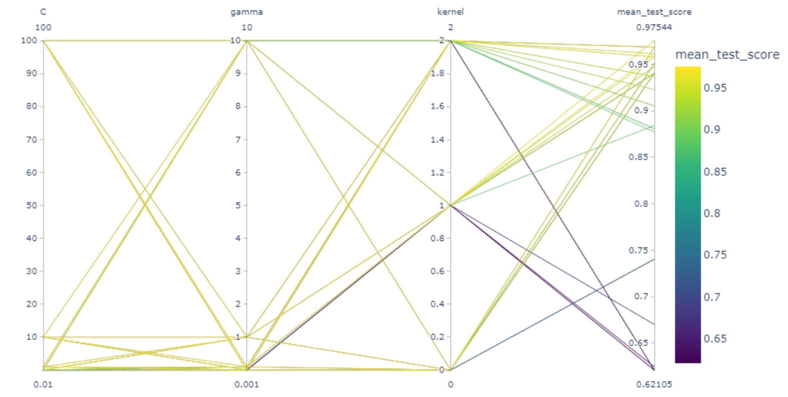


(a) Comparison of evaluation methods

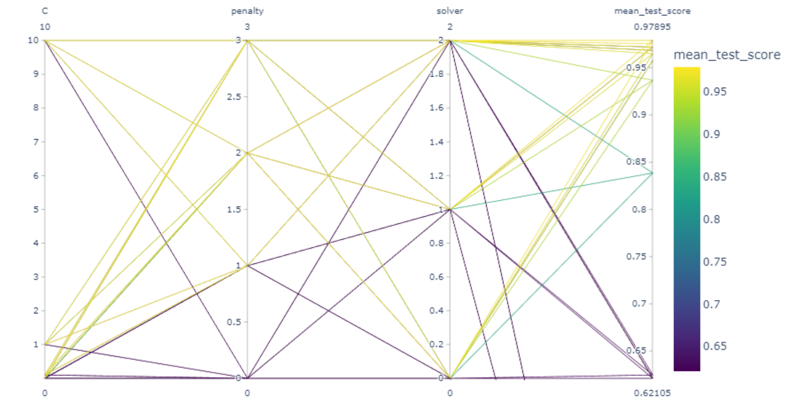


(b) Comparison of scaling methods

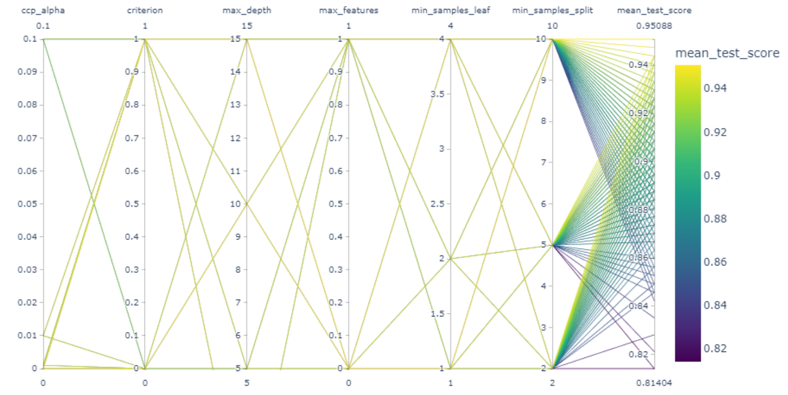
Figure 8: Comparison of different methods on the Breast Cancer dataset



(a) **SVM.** Categorical legend –kernel: 0 → linear, 1 → poly, 2 → rbf



(b) **Logistic Regression.** Categorical legend – penalty: 0 → elasticnet, 1 → l1, 2 → l2, 3 → none; solver: 0 → lbfgs, 1 → sag, 2 → saga



(c) **Decision Tree.** Categorical legend – criterion: 0 → entropy, 1 → gini; max features: 0 → log2, 1 → sqrt

Figure 9: Hyperparameter analysis for Breast Cancer

	F2-score			
	min	mean	max	std
Algorithm				
Decision Tree	0.792079	0.888502	0.945946	0.058440
Logistic Regression	0.922330	0.955152	1.000000	0.028718
SVM	0.922330	**0.958840**	1.000000	0.038636

Table 4: Final comparison with best scaling scheme and tuned hyperparameters for the Breast Cancer dataset

2.2.2 Loan

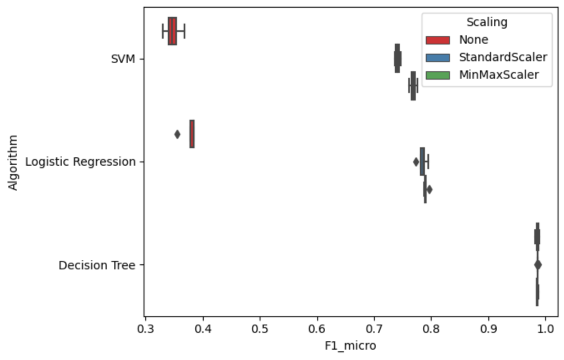
Exploration. First, we analyzed the name of the columns and the type of data they contain. This dataset contains values of type float, int and object. While the numerical values can be analyzed and processed directly, values of type object need to be re-encoded.

Encoding. For categorical data, the one-hot strategy was used, which preserved as much information as possible. Variables taking only two values have been encoded as binary. Finally, we calculated the mean, median and standard deviation of all columns. This informed us about the statistical properties of the features and target data. The target variable has been encoded with the label encoder, which preserved the ordinal characteristic of the grades. The best grade was assigned the number 0, the worst grade the number 6, the other grades have been encoded accordingly. The target data is unbalanced, this informed our choice of metrics.

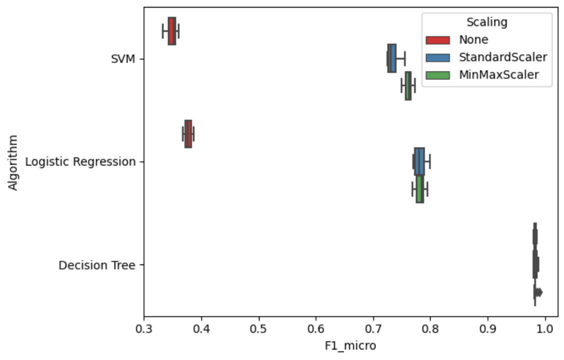
Missing values. We explicitly check every cell for missing values. This set had none.

Performance measures. Since the target data is unbalanced and we wish to have high precision and recall simultaneously, we chose F1 as a metric, with its two variants F1-macro and F1-micro. The first calculates the F1 score for each class and then takes the average across all the classes. This means that each class is weighted equally, regardless of its size or distribution. F1-micro, on the other hand, calculates the F1 score globally by counting the total number of true positives, false negatives, and false positives across all classes. This means that the F1 score is weighted by the number of samples in each class. Given our imbalanced dataset, F1-micro is the most informative metric. We also computed accuracy and the training time necessary for each algorithm. For imbalanced datasets accuracy may be biased towards the class with a larger sample size, but we kept it as a metric so that we could compare the performance of the algorithms across datasets.

Experimental results. We apply both evaluation methods on our classifiers and compare their performance using the F1-micro measure. We also analyse the impact of scaling on the performance. The result of this comparison is shown in Figure 10.



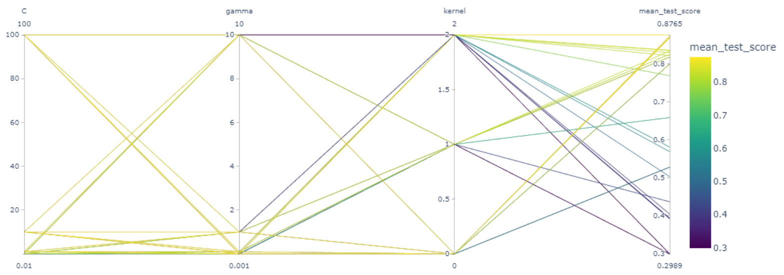
(a) Cross-validation F1 micro



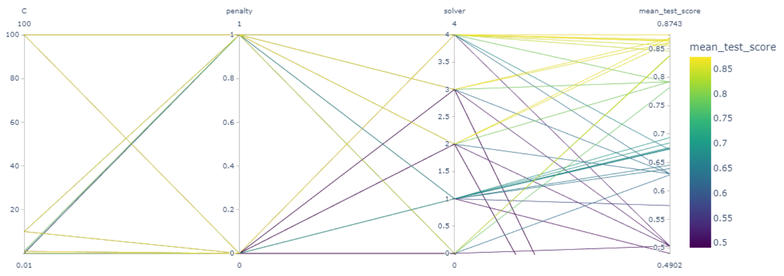
(b) Hold-out F1 micro

Figure 10: Cross-validation vs. Hold-out on the Loans Dataset

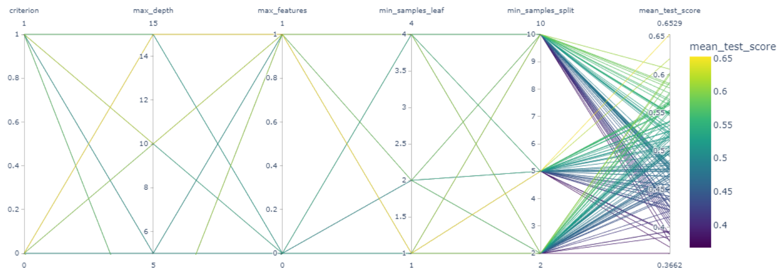
For Cross-validation, SVM has the poorest performance, followed by Logistic Regression and Decision tree, as seen in 10a. For Hold-out, the situation is similar, as seen in 10b. However, the SVM and Logistic regression classifiers show more variance. In both cases, Logistic Regression is most affected by scaling, the Min-Max-scaler improves its performance considerably. SVM is also improved the most under Min-Max scaling. We notice that in both cases, the decision tree has the best performance, with little to no variance, regardless of scaling. This may be explained by the changes made in the pre-processing steps. Lastly, we perform hyperparameter tuning. The results of the grid search for each algorithm are shown in Figure 11, and the final F1-scores of best models per each algorithm are given in Table 5.



(a) **SVM.** Categorical legend –kernel: 0 → linear, 1 → poly, 2 → rbf



(b) **Logistic Regression.** Categorical legend – penalty: 0 → elasticnet, 1 → l1, 2 → l2, 3 → none; solver: 0 → lbfgs, 1 → sag, 2 → saga



(c) **Decision Tree.** Categorical legend – criterion: 0 → entropy, 1 → gini; max features: 0 → log2, 1 → sqrt

Figure 11: Hyperparameter analysis for Loan

	F1-micro-score			
	min	mean	max	std
Algorithm				
Decision Tree	0.9845	**0.9857**	0.9880	0.001440
Logistic Regression	0.7875	0.7907	0.7970	0.003735
SVM	0.7620	0.7688	0.7760	0.005322

Table 5: Final comparison with best scaling scheme and tuned hyperparameters for the Loan dataset

3 Discussion and Conclusions

Preprocessing. Different preprocessing methods seem to work on some algorithms and datasets more than on others. Although it is generally considered unnecessary to scale data for the Decision Tree algorithm, we saw some improvement when doing it. SVM and Logistic Regression seem to be quite sensitive to scale differences.

Algorithms. SVM seems to perform best on binary classification tasks. Although its deviation is a bit larger than the one of Logistic Regression, it is also much faster to train. Decision Tree, on the other hand, seems to thrive when used on datasets such as the Loan, where we have a lot of categorical features.

Hold-out vs cross-validation. Cross-validation makes for a more reliable estimate and efficiently utilizes the available data by repeatedly cycling through different train-test splits. However, hold-out is faster and easier to use. When repeated it can act as an alternative to cross-validation, but because the split is random, it doesn't guarantee effective usage of the data.