Erasmus University Rotterdam

Advanced LESS; using clustering, local regressor sets and global predictor tuning for increased prediction accuracy

Student name
Lorenz Kremer

Student number
532343

Supervisor name
(Utku) U Karaca

Second assessor name
(Hakan) MH Akyuz

Erasmus School of Economics
Bachelor thesis in Business Analytics and Quantitative Marketing

July 3, 2022

Abstract

In this paper we propose three alternative versions of the LEarning with Subset Stacking (LESS) model [Birbil et al., 2021] to improve prediction accuracy. The goal of LESS is to find relations between input and output data with heterogeneous features by training multiple local regressors using subsets of the original data. These local models are then combined into a global predictor. We identified three possible restrictions in the original LESS model. Firstly, subset generation is random. Thus, no optimality is guaranteed. Secondly, local learning is only provided with one regression technique, assuming that every subset is best predicted using the same approach. Lastly, global regression is primarily done using linear regression. We introduce *LESS_cl* which uses a clustering technique as opposed to random subset generation, *LESS_cl_ll* which adds multiple regression techniques to the local learning step, and *Advanced LESS* which extends *LESS_cl_ll* by tuning the global estimator. We compare the performance of our models to various well known regression methods on 10 datasets. We conclude that *Advanced LESS* is the best performing model, outperforming default LESS. It is also competitive with the other methods for most datasets.

Contents

Introduction **3**

Literature **4**

Learning with randomized subsets 5

Local learning methods 5

Data **6**

Methodology **7**

Original LESS 7

LESS extensions 9

LESS example 11

Results **11**

Conclusion **13**

Introduction

If we have a dataset (X, y) with X being the input variable, and y being the output variable, we can use common regression techniques such as linear regression or decision tree regression to create prediction models for y using X. However, if the input data can be split into well-separated subsets, using a regression technique that treats the entire input data as one can lead to lower prediction accuracy. Dividing the original data into multiple subsets and treating these subsets as separate training datasets can significantly increase prediction accuracy.

In this paper, our main goal is to improve prediction accuracy for the LEarning with Subset Stacking (LESS) algorithm as introduced in [Birbil et al., 2021]. The idea behind LESS is to split the dataset into different subsets on which local regressors are trained. Then, a global estimator is trained using features that contain the local estimators and weights which are dependent on the relative position of a sample to each of the subsets.

We analysed the five steps made in LESS and identified some possible restrictions. Firstly, for subset creation [Birbil et al., 2021] mainly focus on an anchor selection approach where clusters are formed randomly. This random method does not maximize inter-cluster distance and will not guarantee well-separated clusters, even though this is crucial for high accuracy prediction [Sreedhar et al., 2017]. Although the paper discusses a LESS variant where K-Means clustering is used, it lacks further clustering applications. Secondly, local learning is done using the same regression technique for every generated subset. Assuming that every subset can be efficiently described using the same type of regression can be seen as significantly restrictive. Even with perfect clustering cluster shapes may greatly differ, and thus, enforcing the same technique on every cluster may come at the cost of overall prediction accuracy. Lastly, the original paper mainly discusses using a linear regressor as global estimator. In the paper, only one example is shown where apart from a linear regressor, also a decision tree regressor is considered.

Apart from changes in the LESS algorithm, we also decided to include three more datasets when testing model performance. Doing this will broaden our understanding of the performance of the models. To answer our main research question "*How can we increase prediction accuracy in the LESS model*", the points mentioned above lead us to form the following sub-questions:

SQ1. *How does LESS prediction accuracy change when using clustering methods for subset*

generation?

SQ2. *How does LESS prediction accuracy change when we provide multiple regression techniques in the local learning step?*

SQ3. *How does LESS prediction accuracy change when using non-linear regressors as global estimators?*

Since the original LESS model does not allow the use of more than one regression technique in the local learning step, we have slightly modified the algorithm to allow for different models to be used on different subsets.

The following outlines how our modified LESS version works. First, we use a clustering method to form datasets instead of using the random anchor approach as was mainly used in [Birbil et al., 2021]. Then, local models are trained using these generated subsets. In contrast to the original LESS model, we allow the user to provide a collection of regression techniques for the local training step. After this, features are generated by combining weights and the local estimators. These weights are inversely related to the data point's distance to each of the local learner's subset. Then, a global estimator is trained using the features. For this step, we consider three different types of regression.

The remainder of this paper is structured as follows: In the literature section we discuss relevant literature that discusses methods similar to the LESS method. In the data section we give an overview of the 10 datasets that we use in this paper. In the methodology section we first present the workings of the original LESS model, followed by an explanation of how exactly we updated LESS for our purpose and an example of our LESS model on 2-d trigonometric data. Then we compare the performance of our updated LESS with the default LESS and some sophisticated prediction methods on the datasets mentioned in the data section. In the conclusions section, we conclude this paper and also mention some potential extensions to our LESS method.

Literature

In this section we discuss literature that is related to our new LESS implementation as well as the original LESS model.

Learning with randomized subsets

There are multiple methods that use learning with randomized subsets. When all of the local predictors are treated the same, meaning that no weights are used for aggregation, this approach is called Bagging [Breiman, 1996a]. Unlike LESS, Bagging uses a bootstrapping method for subset selection so that subsets are generated via sampling with replacement. The size of the newly generated subset is equal to the size of the original subset, but duplicate data points may occur. Subagging [Bühlmann and Yu, 2002, Andonova et al., 2002, Evgeniou et al., 2004] is a version of Bagging where the training dataset is split up into multiple different smaller subsets. In contrast to the Bagging methods, Stacking [Breiman, 1996b, Wolpert, 1992] uses optimization to weight each of the local regressors in the aggregation stage. Magging [Bühlmann and Meinshausen, 2014] is an approach that combines features of both Bagging and Stacking. Particularly, aggregation is done by a weighted average of the local estimators, where generally, the weights are varying. Furthermore, the weights used in their approach do not depend on the response variable.
The LESS method we propose differs from both the original LESS as from the methods mentioned above since it uses samples' locality information for weight generation (as LESS does), and enables users to provide a group of estimation techniques as local regressors, as in [Breiman, 1996b].

Local learning methods

Because of the local learning step in LESS, it can also be seen as a local learning method. There are several methods that rely on local learning. LOcally Estimated Scatterplot Smoothing (LOESS) [Cleveland, 1979] is one of these methods. LOESS applies a simple regression model (often linear) to a small subset of points around the point that is to be predicted. LOESS can be seen as a generalization of kernel regression [Nadaraya, 1964], which fits kernel functions around each known data point. When we want to predict a new data point, we use a combination of the known outputs, where each output is weighed with weights calculated using the kernel functions.
Another similar method is the Gaussian Process Regression (GPR) method [Williams and Rasmussen, 2006]. Here, a Gaussian process is used to evaluate the distance between data points. A covariance matrix is created which has higher values is samples are closer to eachother. When the size of the datasets is very large the covariance matrix may be difficult to

compute. For this scenario, the Bayesian Committee Machine (BCM) [Tresp, 2000] can be used. It splits the dataset into multiple subsets and trains regression systems on these.

Data

In this section we give an overview of the data. In total we used 10 datasets. Of these datasets, 7 were also used in [Birbil et al., 2021], these can be seen in the top half of Table 1. The CPU dataset has the least number of samples, and also contains the least amount of information since the number of attributes is also small. The Superconduct dataset has the most amount of samples and also contains the most amount of data, since it also has the largest number of attributes. All of the datasets used in this paper are from [Dua and Graff, 2017].
In Table 1, Instance gives the name of the dataset, Samples shows the number of samples in the dataset, and Attributes shows the number of explanatory variables in the dataset.

Table 1: The properties of the datasets

Instance	Samples	Attributes
Abalone	4177	8
Airfoil	1503	5
CCPP	9568	4
CPU	209	7
Energy	19735	26
Housing	506	13
Superconduct	21263	81
Fires	244	10
Auto	392	7
Concrete	1030	8

Most of the data attributes were continuous. However, some of the features contained discrete values. For the Abalone dataset, we had to manually transform the $Sex \in \{Infant, Male, Female\}$ attribute into numerical values $\{-1, 1, 0\}$. We removed some of the attributes, such as dates and car names, since these couldn't be efficiently converted to numerical values. From the Energy and Fires dataset, we removed the *date* attributes. From the CPU dataset we removed 2 name attributes and finally from the Auto dataset we removed the *car-name* attribute. The Auto dataset was the only dataset containing missing values. Since we were only dealing with six of such data points, we simply removed them from the dataset.

Methodology

In this section we will first discuss the original LESS model as introduced in [Birbil et al., 2021] and then we will explain the new LESS version where adjustments in subset creation, local model training and global model training were made. We also present an example of our extended LESS model with specific choices for the clustering method and provided local estimators.

Original LESS

We consider a training dataset (X, y), where X contains the input vectors $x_i \in \mathbb{R}^p$. Furthermore, y_i denotes the output of the data point where $y_i \in \mathbb{R}$ with $i \in \{1, ..., n\}$.
The LESS algorithm splits the original dataset (X, y) into multiple subsets (X_j, y_j), $j \in \{1, ..., m\}$ where the goal is to cluster similar data points. After this step, regression models are applied on each of the subsets and the trained models are combined to form features which are used to train the global model.

Subset selection. The first step of the LESS algorithm consists of splitting the original dataset into m different subsets. There are multiple approaches to subset selection. [Birbil et al., 2021] mainly focus on a selection process where m initial anchor data points are chosen at random, and then for each anchor k-nearest samples are chosen and grouped into the same cluster. Alternatively, one can use clustering methods in order to cluster the dataset. Methods like K-means clustering do not only minimize the inter-cluster distance, but also try to maximize the intra-cluster distance [Witten and Tibshirani, 2010]. This leads to better separated clusters and will help increase prediction accuracy.

Local learning. The main idea behind LESS is to train different local predictors for every subset. In the original LESS algorithm, one type of regression is given and the m different subsets are used to train models $\mathcal{L}(x|X_j, y_j)$, $j \in \{1, ..., m\}$. When predicting a new data point, we want to weigh local predictors differently according to their subset's distance to the data point. In other words, we inflate the weight given to local predictors of which the dataset is closer to the data point. To this purpose, we define a weight function that is inversely related to the distance from x_i to subset X_j :

$$w_j(x) = \frac{exp\{-\lambda d(x, \bar{x}_j)\}}{\sum_{j'=1}^{m} exp\{-\lambda d(x, \bar{x}'_j)\}} \tag{1}$$

Here, $d(\cdot, \cdot)$ can be any distance metric chosen by the user and $\bar{x}_j$ is the centroid of the cluster. The centroid may be computed as the average of all data points in a cluster or may be explicitly given for some clustering methods such as K-means clustering. Furthermore, λ is a parameter that determines the influence the distance metric has on the weights. Note that the default weight function chosen for our research is the same as provided by [Birbil et al., 2021].

Feature generation. The third step in LESS is feature generation. This is where we transform our original data (X, y) into feature data points in order to add the local proximity information into the data used to train the global predictor. We do this by combining the locally trained models $\mathcal{L}(x|X_j, y_j)$ and the previously calculated weights $w_j(x)$ in the following manner:

$$z(x_i) = [w_1(x_i)\mathcal{L}(x|X_1, y_1), \dots, w_m(x_i)\mathcal{L}(x|X_m, y_m)]^\intercal \tag{2}$$

These feature vectors contain additional information regarding the relative position of a data point to the clusters.

Global learning. The second to last step consists of learning a regression model using the feature data vectors previously generated. The aim is to use the feature vectors z_i to predict the outcome y_i. To simplify notation, we will use Z to be the matrix containing the feature vectors $z_i^\intercal$ as rows. We denote the trained global model as $\mathcal{G}(z|Z, y)$ so that the prediction of a sample $x_0 \in \mathbb{R}$ becomes $\hat{y}_0 = \mathcal{G}(z_0|Z, y)$ where z_0 is defined by 2.

Averaging. Finally, averaging can be applied when subset creation is random. In the original LESS model random anchor points are chosen. Thus, averaging can be applied to reduce variance. When clustering methods contain a randomization parameter for initial centroid initialization, averaging will also be applied. When averaging is applied through r iterations of the above algorithm, the final prediction will be:

$$\hat{y}_0 = \frac{1}{r} \sum_{\ell=1}^{r} \mathcal{G}(z_0^{(\ell)}|Z^{(\ell)}, y) \tag{3}$$

where $z_0^{(\ell)}$ refers to (2) for iteration ℓ.

LESS extensions

Here we present the adjustments made in our implementation of the model. There are three major changes we made in our version of LESS. The first change appears in the first step where we used clustering algorithms to form subsets instead of the random anchor selection method used in the original model. In the second step we lifted the constraint of using the same type of regression model for every subset and enabled different types of models to be fitted on different subsets. The last adaption made was in step four of the process, where we test the effect of using different types of regression models for the global predictor.

Using clustering methods. Since the random anchor selection technique picks random points as cluster centroids and directly assigns the clusters, there is no guarantee of finding well-separated clusters. Since the separation of clusters is essential for prediction accuracy, considering clustering methods instead of random assignments seems like a logical step. To form clusters, we consider the K-means, Birch, Optics and Mean-Shift algorithms. This means we tested our model using all of these alternatives, and choose the best performing one for each dataset.

K-Means [Sreedhar et al., 2017] begins by choosing m different initial centroids and assigning each data point to the nearest centroid. Once these clusters are formed, the algorithm computes the new centroid of each cluster by averaging over all points in a cluster. Each data point is then again assigned to the nearest centroid and this is done until (local) convergence is reached i.e. the clusters don't change anymore. K-Means is a fairly simple clustering method and is efficient for most datasets with even cluster sizes with flat geometry.

In contrast to the K-Means algorithm, Birch [Zhang et al., 1996] uses hierarchical clustering. Birch builds a height-balanced Clustering Feature Tree which contains information about the data instead of all the actual input data. Since Birch uses a compressed version of the input data, it is very practical for high-dimensional, large datasets.

The Optics algorithm [Ankerst et al., 1999] finds density-based clusters in spatial data. This means that it identifies clusters based on the idea that clusters are continuous regions of

high point-based density, separated by regions of low point-based density. Optics clustering can be used for datasets of uneven cluster sizes, variable cluster density, and it is also good for outlier removal.

Finally the Mean-Shift algorithm is based on Kernel Density Estimation, which estimates the probability density function of the data. As the name suggests, Mean-Shift tries to shift the data points towards the nearest peak of the density function. Mean-Shift uses a bandwidth parameter to evaluate the number of clusters. A large bandwidth will often result in fewer clusters, and a small bandwidth will result in more clusters.

Optimized local and global learning. A large restriction in the original LESS model is that every subset within an iteration of the process is trained with the same type of regression model. If a subset has a non-linear shape and the local estimator happens to be a linear model, the prediction accuracy will be significantly lower. This lead us to remove this constraint by allowing the user to specify a number of different regression types $\mathcal{L}_1(x|X_j, y_j)$. Once the subsets are formed, our algorithm trains each of the provided regression techniques on the first cluster and chooses the technique that produces the lowest mean squares error. It does this iteratively for the clusters, until every cluster has an associated trained model. The models provided for this step are entirely up to the user. However, we mainly focus our efforts on simple linear regression, Kernel Ridge regression and on Decision Tree Regression.

Kernel Ridge regression is a method that combines Ridge regression with the kernel trick. Ridge regression is used when the explanatory variables (X_j) are highly correlated, so that the regular LR parameters would be difficult to compute. The kernel trick uses kernel functions to transform the data into a higher dimension, without actually computing new coordinates. Decision Tree regression uses a regression tree to form the best fitting model. When forming the tree, mean squared error is used to evaluate which decisions to make.
For our last extension, we test model performance when using different regression techniques for the global estimator. Again, we consider linear regression, Kernel Ridge regression and Decision Tree regression.

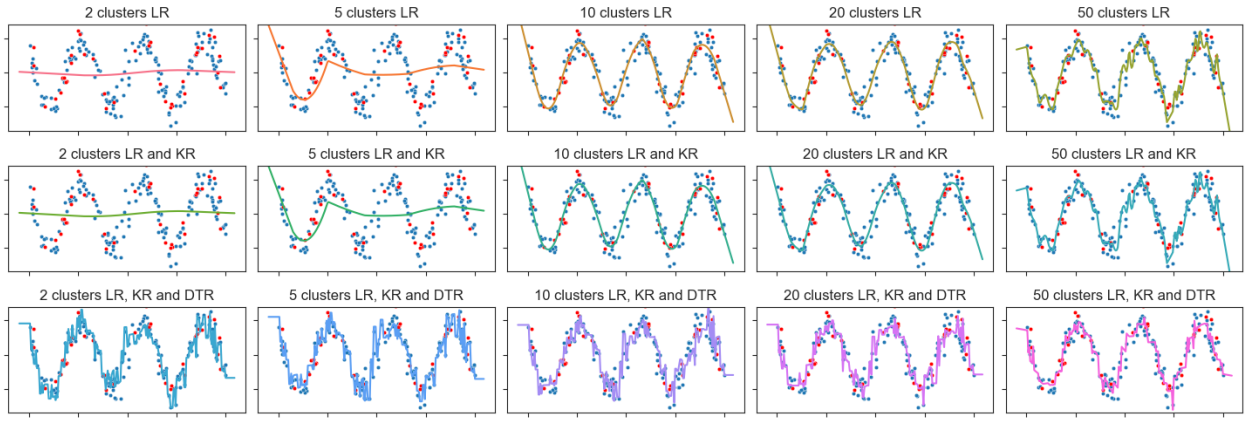

Figure 1: LESS variants using K-Means clustering with varying number of clusters and different local estimators provided. linear (LR), kernel ridge (KR) and decision tree (DTR) regression. Global learning is done with LR

LESS example

Now that we have introduced our version of LESS, we will present an example of how our implementation works on a 2-dimensional dataset generated with a sin function, similar to the approach used in [Birbil et al., 2021]. In Figure 1, the blue dots represent the data points in our training set, and the red dots represent the test set. From left to right, we see the effect of increasing the number of clusters, and from top to bottom we see the effect of added local estimators.

In the top two rows, we clearly see that the fit of LESS highly depends on the number of clusters provided to the K-Means method when using LR and KR. We see that too few clusters lead to a bad fit, but too many clusters may lead to overfitting. The bottom row, where Decision Tree regression is added, we see that the number of provided clusters plays a significantly smaller role. This is because decision tree regressors can find non-linear patterns in the data. We observe an increased variance in the bottom row which could mean that adding the decision tree regressor will cause overfitting.

Results

In this section we compare the performance of seven different models on the datasets mentioned in the Data section. The models we tested are the following. Firstly, our *Advanced*

LESS, which implements all of the possible extensions proposed in the methodology section. This method uses clustering, a set of local learners, and is tuned using different global learners. Next, we propose an algorithm without tuning of the global learner (*LESS_cl_ll*), meaning it uses clustering and a set of local learners. Our third model only uses clustering (*LESS_cl*), and is most similar to the *default LESS*. Furthermore, we test three state of the art regression techniques. We use a Random Forest (RF) approach, a Gradient Boosting (GB) approach and a Support Vector (SVR) approach. We implemented these methods in Python 3.7, and used a similar approach as [Birbil et al., 2021].

The *default LESS* uses linear regression for both local learning and global learning. Furthermore, the *percentage of samples* parameter is used to compute the number of neighbors and the number of subsets. The random anchor selection method is used to form subsets.
Since the *Advanced LESS* and the *LESS_cl_ll* both use a set of local learners, we must specify this set before running our regressions. For both methods, we provided $\{LR, KR, DTR\}$. For all four versions of LESS, we used the distance function $d(x, x') = ||x - x'||_2$ and a λ value of 0.01. The number of replications LESS uses when the averaging step is performed is set to 20. Furthermore, we used 5-fold cross validation for all of our tests. Since all of these methods' performances rely on hyperparameters, we have used hyperparameter tuning for each of them. All of the hyperparameter sets used for tuning can be found in Table 2.

Table 2: Here we see the hyperparameters that were tuned for each of the given regression techniques.

Method	Hyperparameter Sets
Advanced LESS	clustering algorithm, {K-Means, Birch, Optics, Mean-Shift}; number of clusters*, {5, 10, 20, 50, 100, 200, 500}; global estimator, {LR, KR, DTR}
LESS_cl_ll	clustering algorithm, {K-Means, Birch, Optics, Mean-Shift}; number of clusters*, {5, 10, 20, 50, 100, 200, 500};
LESS_cl	clustering algorithm, {K-Means, Birch, Optics, Mean-Shift}; number of clusters*, {5, 10, 20, 50, 100, 200, 500};
Default LESS	percentage of samples, {0.01, 0.05, 0.1, 0.2, 0.3}
Random Forest (RF)	number of estimators, {100, 200}
Gradient Boost (GB)	learning rate, {0.01, 0.1, 0.5}; number of estimators, {100, 200}
Support Vector Regression (SVR)	regularization parameter, {0.1, 1.0, 10.0, 100.0}

* this parameter set is only used when K-Means or Birch is used

In Table 3 we see the results of the models tested on our datasets. Here, we have highlighted the best performing model for each dataset.
We see that exchanging the random anchor method for a clustering method is beneficial i.e. leads to a lower mean squared error (mse) in only 4 of the datasets. However, when we add the additional feature of providing a set of local regressors instead of only using LR, we see a significant decrease in the mse for 9 of the datasets when compared to both the default

model as well as the model using clustering. Finally, the *Advanced LESS* performs best of all of the proposed LESS models. It significantly outperforms all other LESS models for six of the datasets. For CPU, the default model performs best. and for Abalone, Housing and Auto *Advanced LESS* and *LESS_cl_ll* share the lowest mse among the LESS models.
When comparing our the LESS models to the other methods, we see that the GB model performs best on 5 of the 10 datasets. Although *Advanced LESS* does not manage to outperform all models in many datasets, we can observe that the overall performance of *Advanced LESS* is on par with the performance of the best performing algorithm for most datasets.

Table 3: Results for the regression models trained on each of the datasets. Here, the first value in each cell shows the average mean squared error (mse) using 5-fold cross-validation, and the value between brackets shows the std. of the mse.

Problem	***Advanced LESS***	***LESS_cl_ll***	***LESS_cl***	***Default LESS***	**RF**	**GB**	**SVR**
Abalone	**4.8807 (0.0247)**	**4.8807 (0.0247)**	6.2842 (0.3203)	5.9353 (0.3198)	4.9871 (0.0373)	4.9041 (0.0593)	6.6556 (0.025)
Airfoil	4958998.6279 (475951.8814)	6206029.5957 (687151.1974)	13118842.1485 (4681233.8037)	15098250.3463 (866471.2806)	4649020.4174 (126805.2528)	**4233168.8908 (367473.5918)**	31296664.3464 (303804.6635)
CCPP	12.1935 (0.4465)	13.8933 (0.05)	16.3137 (0.0462)	16.2186 (0.019)	12.2955 (0.0826)	**12.185 (0.3412)**	15.9826 (0.0245)
CPU	2628.2948 (1899.8197)	2628.2948 (1899.8197)	6375.5283 (3103.2694)	**955.8218 (682.8133)**	5711.6423 (2757.6616)	4898.0992 (3087.5366)	17174.9187 (541.5846)
Energy*	5801.0326 (114.0565)	7262.9989 (12.4729)	8263.5161 (15.5816)	8443.4848 (56.4698)	**5392.9915 (41.583)**	6519.6993 (134.4414)	8079.727 (25.5353)
Housing	11.074 (2.0173)	11.074 (2.0173)	17.1607 (1.1664)	11.3743 (0.7842)	10.857 (1.1346)	**10.1554 (1.0796)**	14.7122 (3.2726)
Superconduct*	109.9012 (1.1596)	182.5748 (10.1953)	211.761 (2.0709)	285.4954 (4.0851)	**104.412 (0.9839)**	129.3571 (1.898)	183.2393 (1.4789)
Fires	0.0063 (0.003)	0.0073 (0.002)	0.046 (0.0065)	0.0436 (0.0033)	**0.0061 (0.0027)**	0.0135 (0.007)	0.0377 (0.0037)
Auto	**10.0068 (0.6242)**	**10.0068 (0.6242)**	10.1316 (0.524)	10.6324 (0.7373)	10.4322 (0.6126)	10.4929 (0.9028)	10.2279 (0.2066)
Concrete	29.148 (3.2879)	34.2183 (0.9296)	97.1006 (2.4102)	79.5537 (20.9602)	26.3612 (2.4656)	**21.4201 (1.8937)**	39.1584 (6.5979)

* For these datasets the Optics and Mean-Shift clustering algorithms did not converge

Conclusion

Now that we have analysed the results of all different LESS extensions, we can find an answer to each of3 our sub-questions, which will help us answer our main research question. To recap, our sub-questions were the following:

SQ1. *How does LESS prediction accuracy change when using clustering methods for subset generation?*

SQ2. *How does LESS prediction accuracy change when we provide multiple regression techniques in the local learning step?*

SQ3. *How does LESS prediction accuracy change when using non-linear regressors as global estimators?*

To find an answer SQ1 we compare the performance of *LESS_cl* with *Default LESS*. What we see in the results section is that only adding a clustering method to the default LESS method does not necessarily lead to an increase in prediction accuracy (lower mse), and in

six datasets even leads to a significant reduction. This means that simply adding a clustering method does not lead to better performing LESS model.
For SQ2, we take a look at the performance of *LESS_cl_ll*. Now that we are using a set of local regression techniques instead of a single one, we see that prediction accuracy significantly increases (lower mse). *LESS_cl_ll* outperforms both *LESS_cl* and *Default LESS* in 9 datasets. Providing additional regression techniques for the local learning step significantly improves prediction accuracy.
To answer SQ3 we look at the difference between *Advanced LESS* and *LESS_cl_ll*. Logically, we see that *Advanced LESS* always performs at least as well as *LESS_cl_ll*, due to the fact that the global estimator that is used in *LESS_cl_ll* (LR) is also given in the hyperparameter tuning set of *Advanced LESS*. The two methods have the same mse for 4 datasets. For the other 6 datasets, *Advanced LESS* significantly outperforms *LESS_cl_ll*.
Now that we have answered our sub-questions we can try to formulate an answer to our main research problem "*How can we increase prediction accuracy in the LESS model*". Only adding a clustering method to the default LESS algorithm does not increase prediction accuracy. However, both adding a set of regression techniques to the local learning step as well as using different global estimators can significantly increase prediction accuracy.

There are some limitations to our work which could be addressed in following studies. Firstly, we only tune a limited number of hyperparameters for the clustering methods. For Birch, the maximum number of CF sub-clusters could be tweaked. For Optics, the minimum number of samples needed to form a cluster, the maximum epsilon distance and the epsilon parameter could be tuned both for better clustering performance, but also for faster convergence rates. Furthermore, the method for finding the best local regressor on each subset was implemented without splitting the subset into a training and testing set. Splitting the subsets into two may reduce the chance of overfitting. However, this may increase computation time. Of course, one could also test a larger variety of clustering methods, local regressor sets and global predictors to further improve prediction ability.

References

[Andonova et al., 2002] Andonova, S., Elisseeff, A., Evgeniou, T., and Pontil, M. (2002). A simple algorithm for learning stable machines. In *ECAI*, pages 513–517.

[Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

[Birbil et al., 2021] Birbil, S. I., Yildirim, S., Gokalp, K., and Akyuz, H. (2021). Learning with subset stacking. *arXiv preprint arXiv:2112.06251*.

[Breiman, 1996a] Breiman, L. (1996a). Bagging predictors. *Machine learning*, 24(2):123–140.

[Breiman, 1996b] Breiman, L. (1996b). Stacked regressions. *Machine learning*, 24(1):49–64.

[Bühlmann and Meinshausen, 2014] Bühlmann, P. and Meinshausen, N. (2014). Magging: maximin aggregation for inhomogeneous large-scale data. *arXiv preprint arXiv:1409.2638*.

[Bühlmann and Yu, 2002] Bühlmann, P. and Yu, B. (2002). Analyzing bagging. *The annals of Statistics*, 30(4):927–961.

[Cleveland, 1979] Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836.

[Dua and Graff, 2017] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[Evgeniou et al., 2004] Evgeniou, T., Pontil, M., and Elisseeff, A. (2004). Leave one out error, stability, and generalization of voting combinations of classifiers. *Machine learning*, 55(1):71–97.

[Nadaraya, 1964] Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

[Sreedhar et al., 2017] Sreedhar, C., Kasiviswanath, N., and Chenna Reddy, P. (2017). Clustering large datasets using k-means modified inter and intra clustering (km-i2c) in hadoop. *Journal of Big Data*, 4(1):1–19.

[Tresp, 2000] Tresp, V. (2000). A bayesian committee machine. *Neural computation*, 12(11):2719–2741.

[Williams and Rasmussen, 2006] Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

[Witten and Tibshirani, 2010] Witten, D. M. and Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, 5(2):241–259.

[Zhang et al., 1996] Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114.