Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

A Novel Technique for Segmenting Data for Training an Ensemble Regressor

Ahmed O. Bastawisy[1], Sherif F. Fahmy[1,*], Mohamed H. Abd Elazeem[1], Shereen F. Fahmy[2]

[1] College of Engineering & Technology, Arab Academy for Science, Technology and Maritime Transport, Sheraton Al-Matar, Cairo Governate 4471344, Egypt
[2] College of Business and Technology, Arab Academy for Science, Technology and Maritime Transport, Sheraton Al-Matar, Cairo Governate 4471344, Egypt

ABSTRACT

This work presents a novel method for training an ensemble regressor. The main idea of the work is to expose the meta-unit, as well as individual regressors, to all categories of data in the training set. This ensures that the ensemble is able to handle any input it receives because it has seen something like it before. However, in order to do this, we first need to categorize the training data into classes. We perform the categorization by using a random forest and then analysing its output to find similarities in the data. We then train the ensemble using a subset of each category of the data. This ensures that it has no surprises when it works on the actual data. We process the output of the random forest using several different techniques to determine the best way to cluster the data. Our results indicate that using the proposed technique can result in an R2 of 99.9% compared to an R2 of 98.7% for no categorization of the training data of the meta unit. Proving that the concept results in better performance.

Keywords:
Ensembles; ensemble regression; meta-unit; clustering; difference metrics

1. Introduction

This paper presents a novel method for data preparation for training a stacked ensemble regressor [1]. Typically, the ensemble is trained on a large random sample of data and it is hoped that it has seen enough samples to ensure that it operates properly when presented with inputs during actual operation. But there are several issues with this approach, first, if the sample is not large enough, then the ensemble may not see all possible types of data in its input. This is especially true if one of the types of data in the input is rarer than the rest. In this case, the likelihood of it appearing in a small random sample is small. In addition, even if the sample is large, it is still possible for a rare category of data to be either underrepresented or missing in the training of the ensemble.

Therefore, we propose a novel method for training the ensemble that ensures that it sees samples of all the categories of data in the training set available. This means that even under-represented data will be present in the training set to some extent – certainly to a larger extent than

* *Corresponding author.*
E-mail address: fahmy@aast.edu

https://doi.org/10.37934/araset.55.2.113

if we relied on a random sample. It is our contention that such a mechanism will ensure better results by making sure that the ensemble sees no "surprises". The hypothesis that we propose can be stated as follows:

Hypothesis. By training the ensemble on all categories of data in the training set, we ensure that the results of the regressor will be more accurate.

To test our hypothesis, we develop several different methods of categorizing the data in the training set by using several different distance metrics and clustering algorithms, we test all these techniques against each other and against a baseline where no categorization is performed in order to ascertain their relative performance. The purpose of these experiments is not to get the absolute best regression results in the domain being tested, but rather to ascertain the relative performance of the different techniques of training the meta-unit. Getting the absolute best regression results is orthogonal to the purpose of this work, and can be achieved by using more accurate or diverse component regressors or getting more data for training. We consider such issues out of scope of this work. The rest of the paper is organized as follows

1.1 Literature Review

In this section of the paper, we review the literature on the topics most relevant to our work, namely:

i. Ensemble Regressors
ii. Distance Metrics
iii. Clustering Algorithms

1.1.1 Ensemble regressors

In this part of the paper, we overview the concept of ensemble regression and describe the different techniques used in the literature to do this. We conclude the section by specifying the type of ensemble regression we propose to use to test the hypothesis proposed in this work.

Ensemble regression, like ensemble classification, is the use of several different regression techniques – henceforth referred to as base regressors – and combining their results using some technique to obtain a result that is better than can be achieved by any of the base regressors on their own.

There are two benefits to this technique over using base regressors on their own. First, if the base regressors are diverse – i.e., if they are not prone to make errors in the same way – the ensemble can "cancel out" the errors of regressors in certain regions of the dataspace, by giving weight to the regressors that perform well in that region when combining the results. This means that errors can be smoothed out, providing an end result that is better than the base regressors would have been able to produce on their own.

Of course, if the regressors are not diverse, their errors will re-enforce, producing an aggregate result that is worse than individual regressors. Regressor diversity is orthogonal to this work, interested readers can find out more about it here [2].

There are three main methods for ensemble construction, namely:

i. Bagging: Bagging [3], or Bootstrap Aggregating, is a technique used to create ensemble regressors. It works by creating multiple bootstrap samples from the training data and then training a separate regression model on each sample. It typically, but not necessarily,

uses homogeneous weak regression models. Final predictions are obtained by either averaging or taking a weighted average of the results of each individual regression model. Find the weights is in itself an interesting problem. In [4], we show how this can be done using generative adversarial networks (GANs).

ii. Boosting: Boosting [5] is another method for creating ensemble regression models. But in contrast to bagging, the individual regressors are trained iteratively, with each model being trained on the region of the dataspace that was mis predicted by the previous model. This way, it is ensured that no part of the dataspace is left uncovered by some regressor that is good at it. Aggregation is performed by calculating a weighted average of the result of individual regressors with the weight assigned to each regressor dependent on its performance.

iii. Stacking: Finally, we discuss stacking [1]. In this method, we have two stacked layers of regressors. The first layer consists of the base regressors. These regressors may be heterogeneous in the sense of being completely different models, or the same model trained on different data. In all cases, the output of this first layer is considered the input of the second layer. They are, in that sense, "meta-features" that are input to another regressor. This other regressor, which constitutes the second layer of the stack, is the meta-unit mentioned several times in this paper. It is a regressor that uses the output of the regressors in the first layer as its input and is trained to produce the correct prediction based on these input meta-features. This technique is more computationally intensive than the averaging and weighted averaging typically used in bagging and boosting, but it offers the opportunity to produce better results by learning complex, possibly nonlinear, relationships between the output of the base regressors and the correct prediction. It is this type of ensemble that we consider in this work. Choosing the members of the ensemble to ensure good results is an interesting problem, in [6,7] we present one possible method for doing this.

1.1.2 Distance metrics

Before reviewing the literature on difference metrics, we first need to establish what we will be determining the difference for. As will be seen in section 3, we will run our training sample through a random forest, and then generate a probability density function of possible classification of each data point. The way we do this will be further clarified in section 3, but for now it suffices to know that we need to find the distance metrics between probability density functions. Therefore, the rest of this section will survey the techniques used in the literature to find the distance between two probability density functions.

i. Kullback-Leibler (KL) Divergence: This [8-10] is a measure of the distance between two probability distributions, where the KL divergence between two PDFs f and g is given by:

$$Dkl(f \parallel g) = \int_{\square}^{\square} f(x)\, log\left(\frac{f(x)}{g(x)}\right) dx \quad (1)$$

where ‖ denotes "divergence from" and log is the natural logarithm. While this is an easy metric to calculate, it is not, strictly, speaking, a distance metric. For a function to be considered a distance metric, it needs to obey the following rules

- Non-negativity: The distance between any two points must be non-negative, i.e., d(x, y) ≥ 0 for all x and y.
- Identity of indiscernible: The distance between a point and itself must be zero, i.e., d(x, x) = 0 for all x.
- Symmetry: The distance between two points should be the same regardless of the order in which they are compared, i.e., d(x, y) = d(y, x) for all x and y.
- Triangle inequality: The distance between any two points should be less than or equal to the sum of the distances between those points and a third point, i.e., d(x, y) ≤ d(x, z) + d(z, y) for all x, y, and z ,The KL divergence violates the symmetry and triangle inequalities of a distance metric and is hence not a true metric, but rather a dissimilarity measure. We do not use this measure in our work for that reason.

ii. Jensen-Shannon (JS) Distance: This [11,12] distance metric is a symmetric and smoothed version of the Kullback-Leibler (KL) divergence, which overcomes some of the limitations of the KL divergence as a distance metric. Given two probability density functions (PDFs) p(x) and q(x), the JS distance is defined as:

$$Djs(p \parallel q) = \sqrt{\frac{1}{2}Dkl(p \parallel m) + \frac{1}{2}Dkl(q \parallel m)} \quad (2)$$

where m(x) = 1/2[p(x) + q(x)] is the average PDF of p and q, and DKL is the KL divergence. The JS distance is a symmetric version of the KL divergence because it is the average of the KL divergence between p and m, and between q and m. The square root is added for normalization and to ensure that the JS distance satisfies the non-negativity and identity of indiscernible properties of a distance metric. The JS distance has several advantages over the KL divergence as a distance metric. Firstly, it is a true distance metric that satisfies the symmetry and triangle inequality properties. Secondly, it is bounded between 0 and 1, where 0 indicates that p and q are identical, and 1 indicates that p and q have no overlap. Lastly, it is less sensitive to small differences between p and q than the KL divergence, and it has a smoother gradient that makes it more amenable to optimization and numerical computation.

iii. Earth Mover Distance Metric (EMD): The Earth Mover's Distance (EMD) [13-15], also known as the Wasserstein distance, is a metric used to measure the distance between two probability distributions. It was first introduced in the field of computer vision, but has since found applications in other fields such as statistics, information theory, and natural language processing. Assuming that you have two piles of dirt, each representing a probability distribution. The EMD is the minimum amount of work required to transform one pile into the other. In other words, it is the minimum amount of dirt that needs to be moved from one pile to the other, with each movement incurring a certain cost. The EMD is computed by first constructing a flow network between the two probability distributions. Each element of the first distribution is associated with a source node, and each element of the second distribution is associated with a sink node. The cost of moving dirt from a source node to a sink node is given by the distance between the two elements. The goal is to find the minimum cost flow from the sources to the sinks. The EMD has several desirable properties as a distance metric. It is symmetric, meaning that the distance between two distributions is the same regardless of which distribution is treated as the source and which is treated as the sink. It also satisfies the triangle inequality.

iv. Hellinger Distance Metric: The Hellinger distance [16] is a distance metric that measures the similarity or dissimilarity between two probability distributions, and is closely related to the Total Variation (TV) distance.

Given two probability distributions p and q with probability density functions p(x) and q(x), the Hellinger distance between them is defined as:

$$DH(p,q) = \sqrt{\frac{1}{2}\int_{-\infty}^{\infty} \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 dx} \quad (3)$$

The Hellinger distance can be interpreted as the L2 norm of the difference between the square roots of p(x) and q(x), which reflects the geometric distance between their probability density functions in a Hilbert space.

The Hellinger distance has several useful properties as a distance metric. It satisfies the non-negativity, identity of indiscernible, symmetry, and triangle inequality properties, which ensure that it is a valid and consistent measure of dissimilarity between probability distributions. Moreover, it is bounded between 0 and 1, where 0 indicates that p and q are identical, and 1 indicates that they are orthogonal. These are the three-distance metrics we will use in our work.

1.1.3 Clustering algorithms

In this part of the paper, we discuss some of the clustering algorithms that can be used together with the distance metrics discussed in the previous section to divide the data into categories or clusters.

i. K-means clustering: This [17] is a simple and widely used clustering algorithm that partitions data points into k clusters, where k is a user-defined parameter. It works by iteratively assigning data points to the nearest cluster centre (centroid) and updating the centroids based on the mean of the assigned points.
ii. Hierarchical clustering: This [18] is a family of clustering algorithms that build a hierarchy of nested clusters by iteratively merging or splitting clusters based on some similarity metric. There are two main types of hierarchical clustering: agglomerative, which starts with each data point as a singleton cluster and iteratively merges them into larger clusters, and divisive, which starts with all data points in a single cluster and iteratively splits them into smaller clusters. There are many different clustering algorithms available, but these are the two we have selected to test in this work, mainly because of their popularity in the literature and ease of implementation.

Another interesting paper that, while not directly related to time series analysis, gives a very strong overview of the latest models in machine learning techniques is [19]. The paper focuses on brain computer interfacing, but it's a very thorough review of the literature of machine learning. In addition, the authors in [20] propose a very interesting predictive model that incorporates both statistical and machine learning algorithms. Finally, another interesting paper in the field of machine learning is [21], which discusses the use of machine learning algorithms for magnetorheological elastomer carbonyl iron particle concentration estimation.

1.3 Dataset Used

We obtained the closing price of each month for a total of 135 companies listed in the Egyptian stock market from the years 2007 to 2017, we excluded financial companies as their behaviour tends to be different from non-financial companies. The total data collected amounted to about 1500 vectors of stock prices. We frame our regression problem as an attempt to predict one month ahead – i.e., given the closing price for the last 12 months, predict the value of the "13th" month. The system we propose is trained on the data for different companies, over different years. The rationale behind this is to develop a model that can extract general information from the data that can be applied to any company at any point in time. We do not construct a model for each company individually, because, even though this may produce better results, we are attempting to see if we can come up with a general model that models the price behaviour of any company in the Egyptian stock market

2. Methodology

In this section of the paper, we describe the proposed system. As previously mentioned, we are going to use the stacked regression approach. In this approach, there are two layers of regressors, the first consists of the base regressors and the second is the meta-unit that will aggregate the results of the individual regressors. The base regressors will be trained on the same portion of the data set, with their diversity being driven by the fact that they are different – i.e., we are going to use an ensemble regression system with heterogeneous base regressors. The base regressors that we will use in this work are:

i. A fully connected network model
ii. A convolutional neural network model
iii. A support vector regressor model
iv. A random forest models
v. A linear regression models

For both the meta-unit and the individual regressors, we propose a different method of selecting data to train them. Traditionally, they would be trained on a random sample of the training set. The idea is that if the sample is random enough and large enough, it will include all the information that is needed to correctly perform the regression. However, we argue that it is possible to obtain better results by exposing the meta-unit, and the individual regressors, to all examples of possible categories in the dataset. The Hypothesis given in Section I attempts to formalize this claim. In order to do this, we first need to have an idea of what categories exist in the data. There are many different ways to do this, but we chose a method that will allow us to encapsulate the probabilistic nature of the data while categorizing it. Specifically, we use a random forest [22] to divide the data into categories. One way to do this is to simply take the majority vote of the trees in the random forest. This is an easy technique that is not computationally expensive, but we believe that it would not yield good results. Another way would be to consider the vote of each tree in the random forest a vote, and then to construct a histogram of the frequencies of votes (see Figure 1). For example, if there are ten trees in the random forest and four of them vote that the input belongs to class A, then the histogram entry for this particular sample will have a frequency of four in class A. These frequencies can then be normalized to create a probability density function representing the likelihood that that particular sample belongs to a specific class (see Figure 2). To further elucidate this idea, consider Figures 1 and 2.

In Figure 1, four of the trees in the random forest classified the input sample as belonging to Class A, 2 classified it as belonging to Class B, 3 classified it as belonging to Class C and 1 classified it as belonging to Class D.

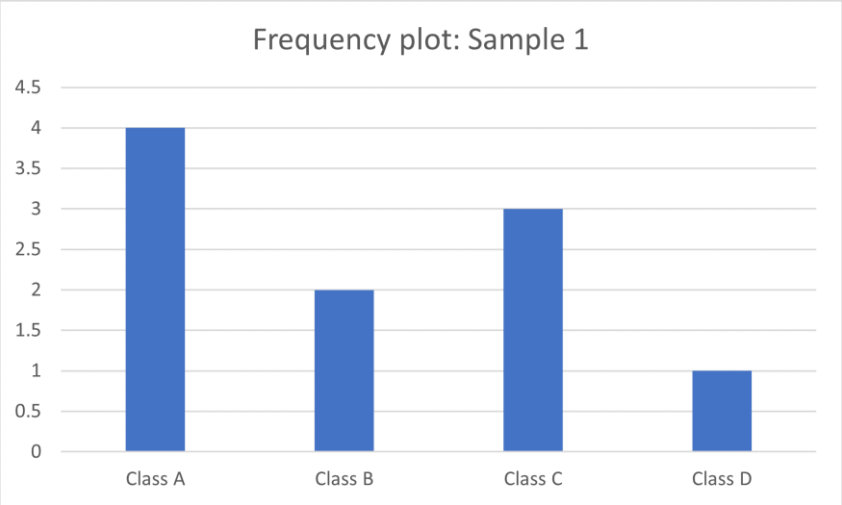

Fig. 1. Frequency plot

Figure 2 shows the result of normalizing the frequency histogram to obtain a probability density function (PDF) for this sample. It is our contention that comparing the PDFs of two samples will provide a better idea of their similarity than a simple majority vote from the random forest. We can state this in a semi-format way as follows:

i. Theorem 1: Comparing the PDFs of the likelihood of two samples belonging to certain classes will provide a better measure of their similarity than using one class label for each sample based on majority votes.
ii. Proof: Consider three samples submitted to a random forest, assume that there are ten trees in the random forest and that they classified the samples as follows:

Sample 1: 4 2 3 1
Sample 2: 3 4 2 1
Sample 3: 6 1 1 2

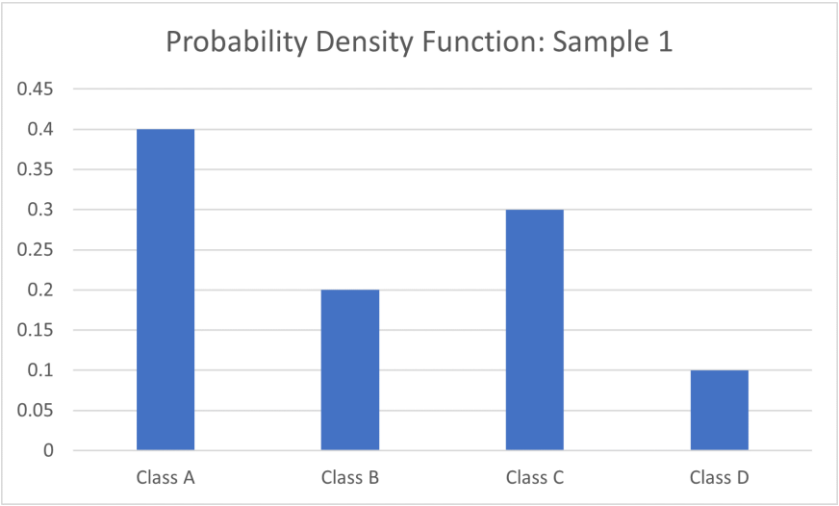

Fig. 2. Probability density function

Consider that the first number is the number of votes for Class A, the second for Class B and so on. This means that a simple majority vote would place sample 1 in Class A, sample 2 in Class B, and sample 3 in Class A. So, a majority vote would place sample 1 and sample 3 in the same category, while placing sample 2 in a different one. It is our intuition that this clustering is incorrect, from the votes of the trees in the random forest it seems that samples 1 and 2 have much more in common with each other than they do with sample 3. It would make more sense to place sample 1 and 2 in the same category, while placing sample three in another. It is this intuition that has motivated us to use the distance between the probability density functions of the likelihood of each sample belonging to certain classes as the metric for clustering the samples rather than just considering the majority vote. Given this basic intuition, we cluster all data according to the similarity of their PDFs, we then use a portion of each cluster as training data. This training data is further subdivided into a portion that is used to train the base regressors and a portion that is used to train the meta-unit. We can summarize the proposed work in the following brief algorithm.

Algorithm 1 Basic Idea

```
procedure SELECTDATAMETAUNIT(data)
    TrainRandomForest(data)
    Run all samples through RandomForest
    FreqHis ⇐ Calculate frequency histogram for all data
    PDFs ← Normalize histograms to obtain PDFs
    Calculate pair-wise distance for all data
    Cluster data using distance above
    Extract T% of each cluster for training
    Extract N% of each cluster above for training meta-unit
```

2.1 The Regressors

In this section of the paper, we will briefly overview the architecture of regressors used in this work. We used grid search to optimize their hyper-parameters.

i. Fully Connected Network The fully connected network used in this work consists of three layers. The first layer contains 20 neurons, with input dimension equal to 12. It is 12 because the network takes the price of a certain stock in the 12 preceding months and outputs the price in the 13th month. All neurons use the relu activation function. The second layer contains 12 neurons, they also use the relu activation function. Finally, the output layer has one neuron with a linear activation function to actually output a stock price. The network was trained using MSE as the loss function and the adam optimizer. It was trained for 1000 epochs.

ii. The convolutional network Our convolutional network has six layers. The first is a convolutional layer and has 60 neurons, with an input size of 12 each. It uses a kernel of size 4 and the relu activation function. The second layer is also convolutional and has 20 neurons, and the rest of its parameters are the same as the first layer. The next layer is a MaxPool layer, with a pooling size equal to 2. This is followed by a flattening layer. After

the flattening layer, we have a fully connected dense layer with 12 neurons that uses the relu activation function. Finally, we have a dense layer with one neuron and a linear activation function to actually output a stock price. The network was trained using MSE as the loss function and the adam optimizer. It was trained for 1000 epochs.

iii. Support vector regressor We used an SVR with C = 1.0, cache size set to 200, epsilon set to 0.4 and the linear kernel.

iv. Random Forest We used a random forest with ten trees, and a maximum depth of 5.

v. Linear Regression We used the standard linear regression model from the sklearn library in python.

vi. The meta-unit Our meta-unit is a fully connected network with three layers. The first has 20 neurons, the second has 12 and the final layer has one. Again, the first two use the relu activation function, while the last uses the linear activation function to actually output a stock price. The same loss function and optimizer is used as in the other neural networks mentioned in this work.

3. Results

In this section of the paper, we describe the experiments conducted and the results obtained. In order to test our hypothesis, we tested several different combinations of distance metrics with clustering algorithms. There are many permutations, and listing them all here would needlessly expand the size of this submission. So instead, we will focus on the three best combinations of clustering algorithms and distance metrics used. These techniques are:

i. Hellinger distance metric with agglomerative clustering
ii. Hellinger distance metric with KNN clustering
iii. Jensen-Shannon distance metric with agglomerative clustering

These combinations produced the best results. In order to reach those results, we first ran the distance metrics mentioned in this paper. We then ran them with grid search hyperparameter optimization and recorded the silhouette scores. We recorded data at 0.6, 0.8 and 0.9 silhouette scores for each of these algorithms. And then we tested each of these combinations with the different clustering algorithms. Thus, we tested three distance metrics, combined with two clustering algorithms at three levels of silhouette scores for a total of 18 different partitions of the data for training the individual regressors, training the meta-unit and testing. We also tried affinity propagation [23], but it did not produce very good results in our case, so we discarded it early in our research. For the Hellinger distance metric with agglomerative clustering, the best result was obtained with a silhouette score 0.6. Here is a table summarizing the results from this score. The table contains the sum of squared errors and the r^2 of the five individual regressors as well as the result of the ensemble. As can be seen in Table 1, the ensemble greatly improves results. It drives r^2 to almost one, we had to use three decimal places to be able to show it without rounding to one. You will also notice that the sum of squared errors went down to single digits. Next, we present the results of the Hellinger distance metric used with KNN clustering. For this combination, the best results were obtained at the 0.8 silhouette score. Table 2 summarizes the results for this experiment. Finally, we present the results for the Jensen-Shannon distance metric combined with agglomerative clustering. The best results for this combination were obtained at the 0.8 silhouette score. Table 3 depicts the results for this experiment.

Table 1
Results for Hilliger vs Agglomerative clustering at 0.6 silhouette

Regressor	Convolutional	Fully connected	Linear regression	Random forest	SVR	Ensemble
Sum of squared errors	474	389	474	1105	359	7
r^2	0.9	0.927	0.904	0.694	0.93	0.999

Table 2
Results for Hilliger vs KNN clustering at 0.8 silhouette

Regressor	Convolutional	Fully connected	Linear regression	Random forest	SVR	Ensemble
Sum of squared errors	479	702	453	809	361	5
r^2	0.887	0.809	0.896	0.763	0.919	0.999

Table 3
Results for Jensen-Shannon vs Agglomerative clustering at 0.8 silhouette

Regressor	Convolutional	Fully connected	Linear regression	Random forest	SVR	Ensemble
Sum of squared errors	364	566	443	640	347	13
r^2	0.920	0.859	0.938	0.837	0.924	0.998

These are the best results obtained from the proposed method. As mentioned before, we conducted several experiments, but the three above produced the best results. Let us now look at the result of the baseline method that does not use categories, but rather hopes that if the training set is large enough it will contain all relevant examples. Table 4 shows the results obtained for this experiment. It is obvious that for the results of the ensemble, the proposed method produces better results. In terms of r^2 the proposed algorithm is always better. The sum of squared errors for the Jensen-Shannon vs Agglomerative Clustering is slightly higher, but this is only one method, the other two produce better results in both parameters – r^2 and sum of squared errors. However, it is interesting to note two things. First, the sum of squared errors in the no categories case is better than in the proposed method for individual regressors that make up the model. While the opposite is the case for r^2. We believe this is due to the fact that the location of errors occurs in different places. That is, in the proposed method, the individual regressors make errors for stocks with large values, this means that a relative error in them will result in larger sum of squared errors. The no categories method appears to make errors, for individual regressors, in low valued stocks. Thus, resulting in lower sum of squared errors. But the higher r^2 implies that the proposed individual regressors are still explaining the variability in the data better. Of course, the ensemble in the proposed method produces much better results than the no categories method. In fact, the ensemble in the no categories method does not provide much improvement on, for example, the convolutional individual regressor. This reinforces the hypothesis of this paper. Specifically, we get a better meta-unit when it sees, during training, all examples of data that it may encounter when running. In the python code for our work, we plot the actual values of the stock's vs the predicted values of the stocks for each of the individual regressors as well as for the ensemble regressor. In order to save space, we will depict only two of them here. The two plots displayed will be one of the individual regressors using the proposed method and the ensemble regressor using the proposed method. The purpose of these two charts is to show how the ensemble improves on the individual regressors and very closely matches the actual value of the stocks.

Table 4
Results for no categories

Regressor	Convolutional	Fully connected	Linear regression	Random forest	SVR	Ensemble
Sum of squared errors	9	27	15	20	13	9
r^2	0.987	0.956	0.979	0.971	0.981	0.987

As can be seen, Figures 3 and 4 show the results of Random Forest individual regressor and the ensemble regressor for the Hillinger vs Agglomerative experiment at 0.8 silhouette score. Notice the blue spikes in Figure 4, especially in the large spike slightly after 40 on the x-axis. The difference between the blue spike and the orange spike is what is contributing to the value of the sum of squared errors for this regressor. Compare it to Figure 4 where the difference between the predicted and actual values are barely visible. Similar graphs are present for all the experiments we conduct, they all follow the same pattern as the charts displayed here.

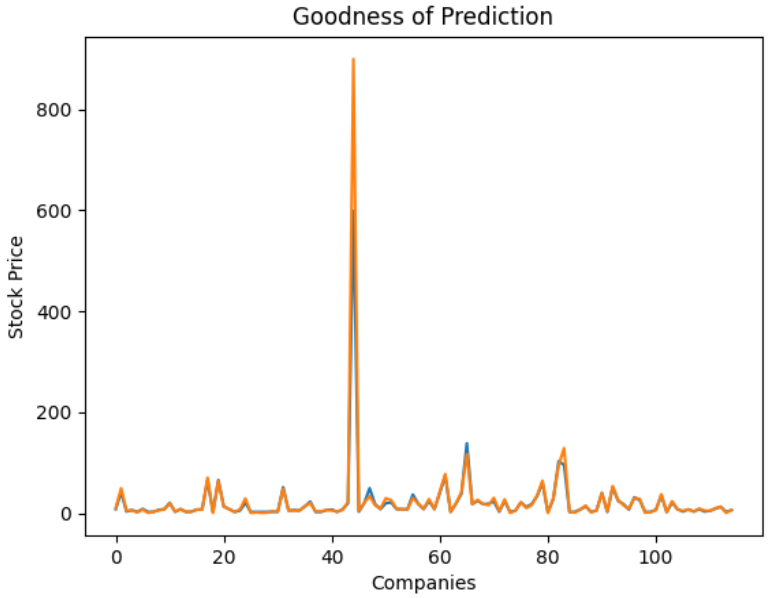

Fig. 3. Random forest, Hilliger vs Agglomerative at 0.8 silhouette

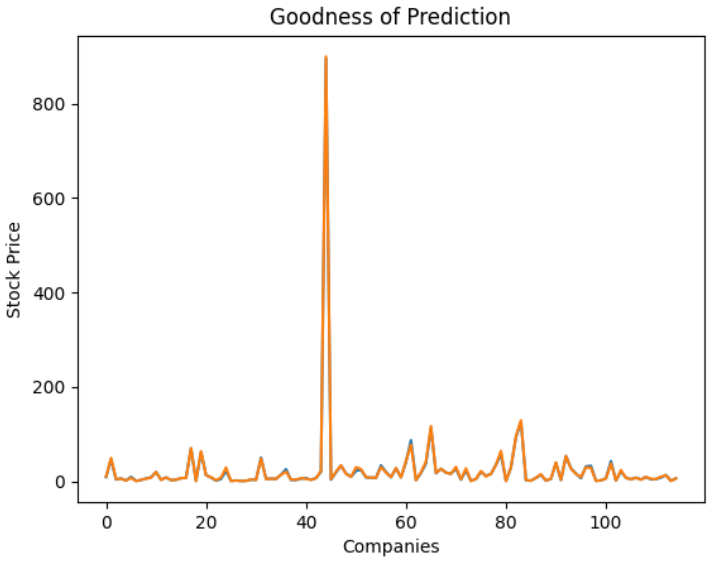

Fig. 4. Ensemble, Hilliger vs Agglomerative at 0.8 silhouette

4. Conclusion

In this work, we proposed a novel method for training ensemble regressors. The novelty lies in the way we partition the data set into categories. The purpose of this partitioning is to make sure that all the individual regressors, as well as the meta-unit in the ensemble regressor, see all the categories of data that they might be exposed to during testing. We do this by proposing a novel method for clustering the data based on the probability density function of their belonging to certain categories as generated by a random forest. The proposed method uses distance metrics to calculate the distances among the probability density functions of each point in the data-set. We then cluster according to different clustering algorithms, and make sure that all regressors see examples from each cluster during training. Our results indicate that the proposed method produces better results in terms of both r^2 and sum of squared errors, indicating that exposing the regressors to all categories of data produces better results. In the future, we propose to test the proposed algorithm on more complex data-sets. In particular, we would like to see the behaviour of the proposed algorithm in the presence of unbalanced data – we believe it will perform even better as it will ensure that scares categories are well represented in the training and testing data.

Acknowledgement
This research was not funded by any grant.

References

[1] A. B. Kahng, W. W. Hwu, and M. S. Yousif. "The next golden age of computer architecture: A plethora of heterogeneity-driven challenges," *IEEE Design & Test*, vol. 29, no. 2, (2012): 6-17.
[2] Zhou, Zhi-Hua. *Ensemble methods: foundations and algorithms*. CRC press, 2012. https://doi.org/10.1201/b12207
[3] Breiman, Leo. "Bagging predictors." *Machine learning* 24 (1996): 123-140. https://doi.org/10.1007/BF00058655
[4] Yehia, Asmaa, Sherif Fahmy, and Ahmed Fahmy. "Generative Adversarial Networks as Weight Generators in Ensemble Classification." In *2023 4th International Conference on Artificial Intelligence, Robotics and Control (AIRC)*, pp. 1-5. IEEE, 2023. https://doi.org/10.1109/AIRC57904.2023.10303257
[5] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of statistics* (2001): 1189-1232. https://doi.org/10.1214/aos/1013203451
[6] Fahmy, Hesham Ahmed, Sherif Fadel Fahmy, Alberto A. Del Barrio García, and Guillermo Botella Juan. "Emotion detection in infants using an ensemble classifier with a novel member selection technique." In *Proceedings of the 2020 Summer Simulation Conference*, pp. 1-9. 2020.
[7] Fahmy, Hesham Ahmed, Sherif Fadel Fahmy, Alberto A. Del Barrio García, and Guillermo Botella Juan. "An ensemble multi-stream classifier for infant needs detection." *Heliyon* 9, no. 4 (2023). https://doi.org/10.1016/j.heliyon.2023.e15098
[8] Cover, Thomas M. *Elements of information theory*. John Wiley & Sons, 1999.
[9] Kullback, Solomon, and Richard A. Leibler. "On information and sufficiency." *The annals of mathematical statistics* 22, no. 1 (1951): 79-86. https://doi.org/10.1214/aoms/1177729694
[10] Bishop, Christopher M. "Pattern recognition and machine learning." *Springer google schola* 2 (2006): 1122-1128.
[11] Lin, Jianhua. "Divergence measures based on the Shannon entropy." *IEEE Transactions on Information theory* 37, no. 1 (1991): 145-151. https://doi.org/10.1109/18.61115
[12] Endres, Dominik Maria, and Johannes E. Schindelin. "A new metric for probability distributions." *IEEE Transactions on Information theory* 49, no. 7 (2003): 1858-1860. https://doi.org/10.1109/TIT.2003.813506
[13] Rubner, Yossi, Carlo Tomasi, and Leonidas J. Guibas. "The earth mover's distance as a metric for image retrieval." *International journal of computer vision* 40 (2000): 99-121.
[14] Leordeanu, Marius, and Martial Hebert. "A spectral technique for correspondence problems using pairwise constraints." In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1482-1489. IEEE, 2005. https://doi.org/10.1109/ICCV.2005.20
[15] Pele, Ofir, and Michael Werman. "Fast and robust earth mover's distances." In *2009 IEEE 12th international conference on computer vision*, pp. 460-467. IEEE, 2009. https://doi.org/10.1109/ICCV.2009.5459199

[16] Jaynes, Edwin T. *Probability theory: The logic of science*. Cambridge university press, 2003. https://doi.org/10.1017/CBO9780511790423
[17] Domingos, Pedro. "A few useful things to know about machine learning." *Communications of the ACM* 55, no. 10 (2012): 78-87. https://doi.org/10.1145/2347736.2347755
[18] Klein, Shmuel Tomi. *Basic concepts in algorithms*. World Scientific, 2021. https://doi.org/10.1142/12298
[19] Talha, Ahmed Zakaria, Noureldin S. Eissa, and Mohd Ibrahim Shapiai. "Applications of Brain Computer Interface for Motor Imagery Using Deep Learning: Review on Recent Trends." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 40, no. 2 (2024): 96-116. https://doi.org/10.37934/araset.40.2.96116
[20] Muhammad, Noryanti, Mohamad Nadzman Mohd Amin, and Rose Adzreen Adnan. "Predictive Modelling on Competitor Analysis Performance by using Generalised Linear Models and Machine Learning Approach." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 45, no. 1 (2025): 51-59. https://doi.org/10.37934/araset.45.1.5159
[21] Saharuddin, Kasma Diana, Mohd Hatta Mohammed Ariff, Irfan Bahiuddin, Nurhazimah Nazmi, Mohd Azizi Abdul Rahman, Mohd Ibrahim Shapiai, Fauzan Ahmad, and Sarah'Atifah Saruchi. "A Comparative Study on Various ANN Optimization Algorithms for Magnetorheological Elastomer Carbonyl Iron Particle Concentration Estimation." *Journal of Advanced Research in Micro and Nano Engineering* 16, no. 1 (2024): 124-133. https://doi.org/10.37934/armne.16.1.124133
[22] Breiman, Leo. "Random forests." *Machine learning* 45 (2001): 5-32. https://doi.org/10.1023/A:1010933404324
[23] Frey, Brendan J., and Delbert Dueck. "Clustering by passing messages between data points." *science* 315, no. 5814 (2007): 972-976. https://doi.org/10.1126/science.1136800