Predictive Model Explainability and a Better Sampling Technique Using SHAP

Aviva Simchon, Osnat Drien

March, 2022

Tabular Data Science - 89-547

Amit Somech

Abstract

In our work, we study the explainability of the ML field and more particularly the SHAP tool focusing on test a new way for sampling the data using *K-mean* clustering, in order to reduce the runtime of the application and which has shown promising results. In addition, we create an applicable system for SHAP in order to plot some of the usable graphs for explaining the prediction for a user with no knowledge in SHAP.

Problem Description

Artificial Intelligent algorithms and Machine Learning models often are perceived as black boxes making inexplicable decisions and are not suitable in different domains, stretching from retail and banking to medicine and healthcare Unlike traditional software, it may not be possible to point to any "if/then" logic to explain the outcome of a machine learning model to a business stakeholder, regulator, or customer. This lack of transparency can lead to significant losses if AI models – misunderstood and improperly applied – are used to make bad business decisions. This lack of transparency can also result in user distrust and refusal to use AI applications. Explainability is the concept that a machine learning model and its output can be explained in a way that "makes sense" to a human being at an acceptable level.

Today there are multiple tools for Explainability such as: SHAP, Lime, Skater, eli5 and others, our project focuses on the SHAP explainability tool. For each of the explainability tool that exists, included SHAP, in addition to ML knowledge the user needs in order to train a predictive ML model, he need to study and understand the field of explainability of the model, and the specific functions and abilities of the tool he uses. We intend to provide an explainability module using SHAP which accessible and simplify the usage of SHAP for the user.

SHAP contains several explainable model types (e.g: linear regression, logistic regression and etc.), we use tree explainers and support three types of tree models. we choose to work with tree explainers since they are faster in run time and more accurate in their results.

SHAP uses the Shapley value from the field of Game Theory to calculate the importance of each feature of the predictive model. The Shapley value is the average marginal contribution of a feature value across all possible coalitions of the feature, so calculate the value requires a lot of computing time. In 99.9% of real-world problems, only the approximate solution is feasible. Usually, the user sends all the dataset to SHAP, then the execution time of explaining all the data is very long. Today the user may use a sample of the data generated randomly, uniformly with the hope of representing the full dataset. We would like to test a different type of technique using SHAP value to generate that sample when our goal is to generate the smallest sample to reduce the execution time and still represents the full dataset.

Solution Overview

Our solution start with finding the smallest and still represented sample of the dataset to explain he model and reduce the execution time. Then, we build an applicable module using SHAP, using the sample we created before. Among the ML model, our module support three types of tree model: CatBoost, XGBoost and tree model of SKLEARN.

Sampling

We look for the smallest size of sample from the data which represent the full dataset using SHAP. While today the samples are generated randomly from the data, We suggest a new way for generating the samples, which was never tried before in this field - *K-mean clustering*.

K-mean clustering is a type of unsupervised learning and one of the popular methods of clustering unlabelled data into k clusters. Our idea of sampling via *k-mean clustering* is splitting the data into k clusters, and then sample from each cluster the relative chosen sample size. By that we gain from each type categorize cluster the representation which eventually all together catch the full represented data.

One of the trickier tasks in clustering is identifying the appropriate and optimal number of clusters k. A commonly used method for finding optimal k value is Elbow Method, which we used. In the Elbow method, we are actually varying the number of clusters (k) from 1 – 10. For each value of k, we are calculating WCSS (Within-Cluster Sum of Square). WCSS is the sum of squared distance between each point and the centroid in a cluster. When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases, the WCSS value will start to decrease. WCSS value is largest when $k = 1$. When we analyze the graph we can see that the graph will rapidly change at a point and thus creating an elbow shape. From this point, the graph starts to move almost parallel to the X-axis. The K value corresponding to this point is the optimal k value or an optimal number of clusters. After accomplish the optimal size of k, we use *K-mean clustering* to sample the data as described above.

Next, we calculate the SHAP value of the full dataset and the SHAP value of the sample we created to compare between them. By calculate the SHAP value we get an $N \times M$ matrix when N is the feature and M is the size of observations you sent to SHAP. In order to compare between the SHAP value to evaluate their similarity, for each of the SHAP value, we average the value of all the values in each observation in a feature and normalized it. Now we have a single normalized SHAP value for each feature for the full dataset and for the sample we created. We next find MAX_F - the feature with the maximum SHAP value in the full dataset and calculate MAD (*Maximum absolute difference* n.d.) value which is the maximum error between the feature value from the full dataset and the sample. Eventually we return *relative MAD*: MAD/MAX_F - which represent the relative error and the best SHAP feature value. When *relative MAD* is small, its mean that the sample is closest to the full dataset.

SHAP Applicable System

Our module get a predictive model which was build from the train data and its name, in addition the user should supply the data he used to build the predictive model. Thus he will understand the way the predictive model was build, internal tendencies of the model, what value/values cause the model to increase or decrease the prediction, The importance of the various features, understanding the distribution of SHAP values or feature combinations, including insights of the data. We supply the sample data we generate in the previous phase. Thus we can examine samples of *false negative* or *false positive* and analyzed the model error with the suitable plots.

Working with our module provides several different graph plots to explain the data. The user can choose to supply arbitrarily an index list of the observations he would like to explain. Alternatively, we choose the top-10 observations from the observation list.

Our module export into a folder name *SHAP* a new folder named by the predictive model with four folders in it:

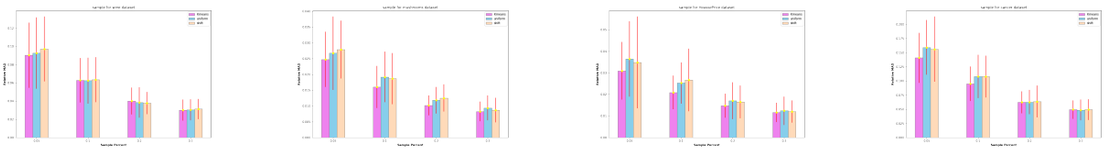
SummaryPlot contains 2 graphs for global understanding of the predictive model results. The first is a bar type (shown on fig 2a), which creates a global feature importance plot, where the global importance of each feature is taken to be the mean absolute value for that feature over all the given samples. The second is a csattered type (shown on fig 2b), which combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The color represents the value of the feature from low to high. Overlapping points are jittered in y-axis direction, so we get a sense of the distribution of the Shapley values per feature. The features are ordered according to their importance.

WaterfallPlot (shown on fig 2c) contains a bar graph for each observation we passed. In each observation graph the bottom of a waterfall plot starts as the expected value of the model output, and then each row shows how the positive (red) or negative (blue) contribution of each feature moves the value from the expected model output over the background dataset to the model output for this prediction.

ForcePlot (shown on fig 2d) contains a bar graph for each observation we passed. Each observation graph demonstrate how each value made our prediction differ from the baseline and allow you to see how features contributed to the model's prediction for a specific observation. This is perfect for being able to explain to someone exactly how your model arrived at the prediction it did for a specific observation.

BarLocalPlot (shown on fig 2e) contains a bar graph for each observation we passed. Each observation presents its SHAP values into the bar plot function creates a local feature importance plot, where the bars are the SHAP values for each feature.

The results explained the data in 3 dimensions: first is the color - the higher the value of the feature the color will be red while the low feature is blue. Second - the features, which arranged according to their importance in the model. Third - the SHAP value, when a low SHAP value lowers the prediction and a high SHAP value raises the prediction.



(a) Wine (b) Mushrooms (c) Boston House (d) Cervical Cancer

Figure 1: *K-mean -relative MAD*, Uniform and Shift with 50 Iterations.

Experimental Evaluation

We next describe our experimental settings, and the results follow. We have used four benchmarks:

Risk factors of cervical cancer (*Risk factors of cervical cancer* n.d.) The dataset was collected at 'Hospital Universitario de Caracas' in Caracas, Venezuela. The dataset comprises demographic information, habits, and historic medical records of 858 patients. Several patients decided not to answer some of the questions because of privacy concerns (missing values).

Wine dataset (*Wine dataset* n.d.) Include two datasets, related to red and white vinho verde wine samples, from the north of Portugal. The datasets in size of 1599 records.

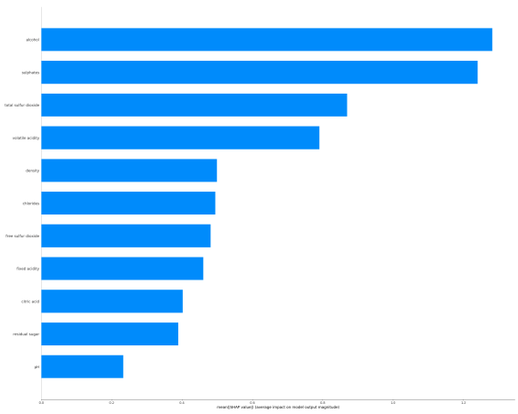
Boston house pricing (*Boston house pricing* n.d.) This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. The dataset in size of 2919 cases.

Mushroom (*Mushrooms dataset* n.d.) This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This dataset contain 8124 instances.

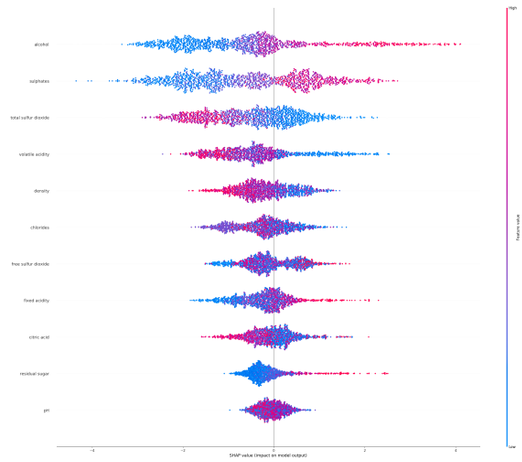
Preprocessing In order to work with each dataset we had to process it (e.g: fill the "null" fields) in order to make the model work with the dataset. As we know that when we split our data for train/test we should first split the data, process the train, and do the same process later over the test. Since our module should get from the user the model and the dataset, we assume that The preprocessing of the data is beeing done by him, and thus, in our project, we was not strict with that. On GIT appear our datasets after preprocessing.

Sampeling We choose to evaluate our *K-mean* sample vs. the traditional uniform random sample and the shift sample which is similar to *K-mean* as they both centroid-based algorithms. The shift sample works by updating candidates for centroids to be the mean of the points within a given region and we do not have to supply the K parameter which make it simpler.

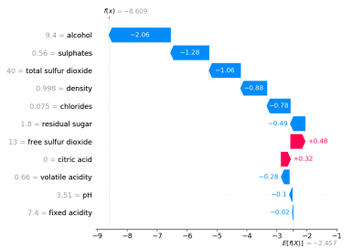
In figure 1 we can see the results for each of the datasets we tested, when we sample in each experiment different size of the data to observe the similarity of each size over the full dataset. We tested 0.05% , 0.1%, 0.2% and 0.3% of the data each time. Consistently, our *K-mean* sampling gets better results when the sample size is up to 0.2%.



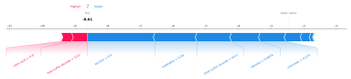
(a) Global Bar plot



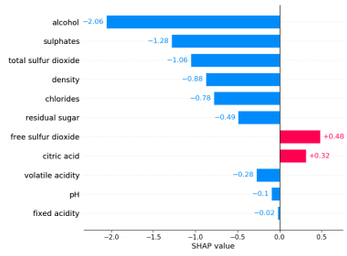
(b) Global Scattered plot



(c) Local Waterfall plot



(d) Local Force plot



(e) Local Bar plot

Figure 2: Plot examples.

Module As for our applicable system we create a predictive model for each of the datasets above. In our example shown on figure 2, we did not sent a specific observation list, and thus generate the plots for the top-10 observations. As described above, our module exports different plot explaining the data. The name of each plot had in its prefix a number, represent the index of the observation (e.g: "5_waterfall_plot_pos.png" mean that the index is 5), then the name of the plot we export.

The plots we present are exports from the 'Wine' dataset. The first line show tow global plots over the hole dataset and the second row show three local plots, of a specific observation. Figure 2a is the Global bar plot, which creates a global feature importance. As for the 'Wine' dataset, the 'alcohol' feature is the most important and its value affects the most on the prediction. In contrast, the 'pH' feature is the less important and affect the least on the prediction. Figure 2b is the Global scattered plot. As in the global bar plot, the features are sorted from the most influence feature to the least influence feature. For each feature on the scale, appear all the SHAP values of all the observations. For example, for the 'alcohol' feature, The SHAP value of an observation is positive (in red) and on the right side of the baseline, which means it has a high value and contributes to the prediction. On the left, we see the negative SHAP values of observations (in blue) and as much they are on the left side, they lower.

Figure 2c, 2d and 2e show the fifth observation from the data, and explain from the base line (which is the average of all predictions), how the prediction was concluded. Each feature get a negative (in blue) or positive (in red) value which represent how much each it contribute this prediction. For example, on 2e, for the fifth observation, 'alcohol' reduce the prediction in 2.06 and in contrast, 'free sulfur dioxide' increase the prediction in 0.48.

Related Work

Over the internet, there are many tutorials and examples of using SHAP such as *SHAP documentation* n.d. - which is the most popular documentation of SHAP, and *Shapley Values* n.d. - which is another documentation over ML and several explainability tools. Still, in order to get the results and understand them properly the user needs more than just formal tutorials. During our research, we study from many other sites in order to deeply understand how to use SHAP and the explanations it export. *K-mean* (*A Simple Explanation of K-Means Clustering* n.d.) is a popular unsupervised learning algorithm, and is being used in different fields. As much as we search, using k-mean for sampling data using SHAP has never been done before.

Conclusion

Although our flow in the project is first, to create the best sample for SHAP and then use it in our module and get the model explanation, our work was the opposite. At first, we study SHAP and understand how SHAP is working, how to work with it, and understanding the different plots, which took a lot of time and amplify the need of our module. Our module supplies the user summary plots and the explanation for each in one place.

The idea for the second part of our project comes from our own usage with SHAP, by using a large dataset that took a long execution time. We suggest generating a sample using *K-mean* and SHAP. During the experiments we see that up to 0.2% of the data, our *K-mean* sample is better representing the whole dataset than the other samples we compare to uniform, and shift. Above that sample size, the difference between it and uniform is not significant and thus we recommend using the uniform sample as it simpler. The shift sample results were the worst all the time and are not recommendable at all.

References

Risk factors of cervical cancer (n.d.). https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+\%28Risk+Factors\%29.

Boston house pricing (n.d.). https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html.

A Simple Explanation of K-Means Clustering (n.d.). https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/.

Maximum absolute difference (n.d.). https://www.geeksforgeeks.org/maximum-absolute-difference-value-index-sums/.

Mushrooms dataset (n.d.). https://archive.ics.uci.edu/ml/datasets/mushroom.

SHAP documentation (n.d.). https://shap.readthedocs.io/en/latest/index.html.

Shapley Values (n.d.). https://christophm.github.io/interpretable-ml-book/shapley.html.

Wine dataset (n.d.). https://archive.ics.uci.edu/ml/datasets/wine+quality.