FEATURE EXTRACTION AND MACHINE LEARNING EXPLAINABILITY THROUGH SHAPLEY VALUES

Michela Cerqua, Ivan Gridelli

Department of Mathematics Tullio Levi Civita

ABSTRACT

In this article we tackled the problems of shedding light over a machine learning's model internal processes and decisions and dimensionality reduction via feature extraction. We used the definition of the Game Theory concept of *Shapley Values*, applied it in different and compared the differences in results. Finally we present an experiment where we apply the methods that we defined.

1. INTRODUCTION

Since the decision making of a machine learning algorithm is treated as a black box, ways to interpret their output have become of utmost importance. Given a database $D = \{(x_i, y_i)_{i=1...n}\}$ where $x_i \in \mathbb{R}^d$ is indexed by d features, we wish to understand which features contributed negatively and which positively to the machine learning's output $f(x)$. Moreover we would like to understand which features were the most relevant in influencing the machine's output. Shapley values exactly embody this concept: they assign a numerical value to each feature proportional to how they changed the output. In section 2 we will go into more detail about how such values are defined. In particular we have developed such applications in the contest of tree-based regression and classification. Then we propose a few different applications in section 3 and compare their behaviour on a specific dataset. In fact, in section 4 we have produced a dataset containing data from [5] and [9] regarding the number of severe road accidents in Italy in 2020 and compared which features (age range, sex *et cetera*) most contributed to the machine's binary output (whether the individual is likely to be involved in an accident or not).

2. THEORETICAL BACKGROUND

Firstly we need to define a few concepts coming from the field of Game Theory.

Definition 1 *A coalition game is a couple (G, v) where G is a set of k players and v is a function from the power set of G i.e. $\mathcal{P}(G)$ to $\mathbb{R}$ that assigns to each coalition the expected payoff of their collaboration.*

Definition 2 *A solutions concept is a function $\varphi : G \to \mathbb{R}$. It can be interpreted as a measure of how valuable is each player to a coalition.*

In particular we are interested in solution concepts with a few desirable properties[7]:

Definition 3 *Dummy Player: If for some player i we have $\varphi(S \cup i) = \varphi(i)$ for all $S \subset G \setminus \{i\}$ then $\varphi(i)$=0.*

Efficiency: $\sum_{i \in G} \varphi(i) = v(G)$

Symmetry: If for some players $i, j \in G$ $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subset G \setminus \{i, j\}$ then $\varphi(i) = \varphi(j)$.

Linearity: If (G, v) and (G, w) are two coalition games then for all $i \in G$ $\varphi_{v+w}(i) = \varphi_v(i) + \varphi_w(i)$ and for all $\lambda \in \mathbb{R}$ we have $\varphi_{\lambda v}(i) = \lambda \varphi_v(i)$.

It can be proven [8] that the only solution concept satisfying all the above mentioned properties is the Shapley value defined for all $i \in G$ as:

Definition 4

$$\Phi_v(i) = \sum_{S \subset G \setminus \{i\}} \frac{|S|!(|G| - |S| - 1)!}{|G|!} (v(S \cup \{i\}) - v(S))$$

This formula is the (weighed) sum of the marginal contributions of player i to each possible coalition S. Intuitively, a negative Shapley value would mean that most of the coalitions that didn't include i performed better, and hence i's importance is very low.

2.1. In the machine learning context

Since our aim is to try to explain how our features impact the machine's decision our set of players G will be the set of features. Since there is no *a priori* best definition for the function v we have defined several different ones and will compare the respective outputs in the final section. To our surprise, since the formulas output was computed in very little time (although requiring at least 2^d computations) no approximation of formula 4 was necessary. However, since the execution time is exponential in the number of features, if one needs to approximate the formula for instance one could sample just a few subsets S as in [3] or use more advanced methods as proposed in [1]. We have developed four different methods to calculate Shapley values. For the first two we have defined custom functions v and we directly evaluate the sum defined in Definition 4. If f is our model, in both cases $v(S)$ represents on average how far $f(x)$ is from the actual value y for $(x, y) \in D_{test}$ where D_{test} is the test part of our dataset. For the second two algorithms we applied a local approach. For each data point we will decompose the machine learning's output into a specific sum that can shed some light over the machine's decision. Averaging out on these local decompositions we can obtain a global perspective. For each point $(x, y) \in D_{test}$ we wish to decompose

$$f(x) = E[f(x)] + \sum_{i \in G} \varphi_i(x)$$

where $\varphi_i(x)$ is i's Shapley value relative to the data point x. In order to do so our formula 4 becomes:

$$\Phi_i(x) = \sum_{S \subset G \setminus \{i\}} \frac{|S|!(|G|-|S|-1)!}{|G|!} (f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S))$$

as seen in [2]. Here f_S is the machine trained only on the features in S and x_S is the restriction of our datum x to only the values relative to the features in S. The task of training 2^d machines, exponential in the number of features, has revealed to be computable in our specific setting hence it didn't require any approximation, although expensive in time. The fourth and final method, based on [6] computes

$$\Phi_i(x) = \sum_{S \subset G \setminus i} \frac{|S|!(|G|-|S|-1)!}{|G|!} (f(\hat{x}_{S \cup \{i\}}) - f(\hat{x}_S))$$

by averaging out $\hat{x}_S$ which coincides with x on features of S and elsewhere is z for all $z \in D_{test}$. In the next section we will go into more detail about the algorithms we implemented.

3. ALGORITHMS

Let us preface that the algorithms presented work for both the cases of regression and classification. For the sake of simplicity we will always assume to be dealing with a machine learning tree f trained as a *DecisionTreeClassifier*. Since our outputs were numbers between one and zero we decided to let the worth function v be between one and zero.

3.1. First Algorithm

Since we directly applied the formula 4, which is quite straightforward, let us just show the definition of the function v.

```
Data: ($D_{test}, S$) where
      D$_{test}$ contains m points
Result: $v(S)$
for $(x_i, y_i)$ in $D_{test}$ do
    $z \leftarrow mean(x_j$ for $(x_j, y_j) \in D_{test})$;
    $z_S \leftarrow x_{i,S}$;
    $diff_i \leftarrow 1 - |f(z) - y_i|$;
end
return mean(diff)
```

Algorithm 1: first v function

Although we will see that this algorithm is quite similar to the second one, we will also observe that it yields slightly different Shapley values.

3.2. Second Algorithm

Once again the formula stays the same hence we will report only the algorithm for the worth function v.

```
Data: $(D_{test}, S)$ where
        $D_{test}$ contains m points
Result: $v(S)$
for $(x_i, y_i)$ in $D_{test}$ do
    for $(x_j, y_j)$ in $D_{test}$ do
        $x_{j,S} \leftarrow x_{i.S}$;
        $diff_j \leftarrow 1 - |f(x_j) - y_i|$;
    end
    $average_i \leftarrow mean(diff)$
end
return mean(average)
```
Algorithm 2: second v function

In section 4 we will see this algorithm's behaviour. The next two algorithms will operate locally, hence we will fix an observation from our dataset (x^*, y^*). Moreover the function v is directly computed in the program that evaluates the Shapley values so there aren't any definitions.

3.3. Third Algorithm

Let us denote by $x \overset{S}{\rightsquigarrow} x'$ the array containing the values of x' in all positions except having the values of x in the positions determined by S.

```
Data: $(D_{test}, (x^*, y^*))$ where
        $D_{test}$ contains m points
Result: Shapley values array
for $feature$ in $G$ do
    for $k$ in $\{0...d\}$ do
        $POW \leftarrow \{S \subset G s.t. |S| = k\}$;
        $C \leftarrow \binom{d}{k}^{-1} \frac{1}{d-k}$;
        for $S \subset POW$ do
            $l \leftarrow \sum_{(x,y) \in D_{test}} f(x^* \overset{S \cup \{i\}}{\rightsquigarrow} x)$;
            $r \leftarrow \sum_{(x,y) \in D_{test}} f(x^* \overset{S}{\rightsquigarrow} x)$;
        end
    end
    $shap_{feature} \leftarrow \frac{C}{m}(l - r)$
end
return shap
```
Algorithm 3: Shapley values of (x^*, y^*) relative to the ML f.

For this algorithm we have that

$$v(S) = E[f(x), x^* \overset{S}{\rightsquigarrow} x]$$

as proposed in [1] which yields Shapley values such that we obtain the following decomposition

$$f(x^*) = E[f(x)] + \sum_{i=1}^{d} \Phi_i(x^*)$$

This decomposition allows us to decompose the relevance of each feature in explaining why the actual prediction was far from the expected output.

3.4. Fourth Algorithm

For this final program we took as a reference the formulae proposed by [6] and [2] and trained 2^d models, one for each possible subset of G.

```
Data: $(D_{test}, (x^*, y^*))$ where
        $D_{test}$ contains m points
Result: Shapley values array
for $feature$ in $G$ do
    for $k$ in $\{0...d\}$ do
        $POW \leftarrow \{S \subset G s.t. |S| = k\}$;
        $C \leftarrow \binom{d}{k}^{-1} \frac{1}{d-k}$;
        for $S \subset POW$ do
            $l \leftarrow \sum_{(x,y) \in D_{test}} f_{S \cup \{i\}}(x^* \overset{S \cup \{i\}}{\rightsquigarrow} x)$;
            $r \leftarrow \sum_{(x,y) \in D_{test}} f_S(x^* \overset{S}{\rightsquigarrow} x)$;
        end
    end
    $shap_{feature} \leftarrow \frac{C}{m}(l - r)$
end
return shap
```
Algorithm 4: Shapley values of (x^*, y^*) using models trained on all possible subsets of features.

As we will see in the next section this model will not be able to globally extract relevant information about our data hence we will focus on the results obtained by our first three algorithms. Notice in particular how our algorithm 3 is an approximation of this model. For this reason and that it is also computationally more expensive we will not delve too much into it in our experiments.

4. EXPERIMENTS

We wanted to inquire about the ”riskiness” of an everyday traffic scenario, so, using data from [5] and [9], we generated a consistent dataset where each row represents a person travelling on a road and six columns which represent the person and road features: the first two columns indicate respectively the person's *age range* and *sex*, the following two are the *vehicle* (pedestrian, bicycle, motorbike, car, bus or truck) and the *role* (pedestrian, passenger or driver), then we have the *road type* (urban, motorway or peripheral) and the *month* of the year. The final column tells us if the person had an *accident* (1 if they did, 0 otherwise).

4.1. Dataset making

First of all, we computed for each feature the percentage of accident involvement. Take a look for instance at the *vehicle* feature

Pedestrian	2%
Bicycle	2%
Motorbike	10%
Car	73.8%
Bus	0.5%
Truck	11.7%

and compare it with the *month* feature:

Jan	7.2%
Feb	6.3%
Mar	7.4%
Apr	8.1%
May	8.1%
Jun	9.8%
Jul	10.4%
Aug	8.6%
Sep	8.5%
Oct	8.6%
Nov	8%
Dec	8%

It's noticeable that, while choosing a different *vehicle* gives us a large percentage difference, the *month* values are very close one to another. This leads us to believe that the *vehicle* choice will be more decisive compared to the *month* choice (i.e. we expect an high Shapley value for the former, and a low value for the latter).
Once we got all these percentages, we generated 200 random persons with random features, making sure that the data we got was plausible (we didn't want a 7-year-old to be driving a truck among our dataset rows), then we generated the last column (*accident*) as follows: a person is represented by a data row $x = (x_0, x_1, x_2, x_3, x_4, x_5)$, where each x_i is the i-th feature, and to each feature corresponds an accident involvement percentage $P_{x_i}\%$. As we noticed before, these percentages are essential to tell whether the person is going to have an accident or not, so we decided to sum them up: $P = P_{x_0} + P_{x_1} + P_{x_2} + P_{x_3} + P_{x_4} + P_{x_5}$. The worst case scenario is when a man (66%) with age between 25 and 44 years-old (32.8%) drives (66%) a car (73.8%) on a urban road (69.6%) in July (10.4%). These values sum up to a total $\tilde{P} = 318.6$, so we set the ”accident threshold” at $P' = 150$ (i.e. roughly $\frac{\tilde{P}}{2}$): if the percentage sum was larger or equal to 150 then the *accident* feature was set to 1 (i.e. the accident does happen), otherwise it was set to 0. We chose this particular P' because we wanted our sample to be representative of the actual traffic accident scenario. We run our dataset generator multiple times and confronted the results with data from [5] and [4] and we found out that with this choice the obtained data was the most accurate portrait of the real situation.

4.2. Feature extraction

Our tree predictor has been trained, using the model *DecisionTreeClassifier*, on the dataset that has been split with 70% for training and 30% for testing.

4.2.1. Algorithms with custom v

For both the first two algorithms we generated 10 new datasets as aforementioned and plotted each feature's Shapley value as depicted in Figure1 and Figure2. Since the dataset was re-generated each

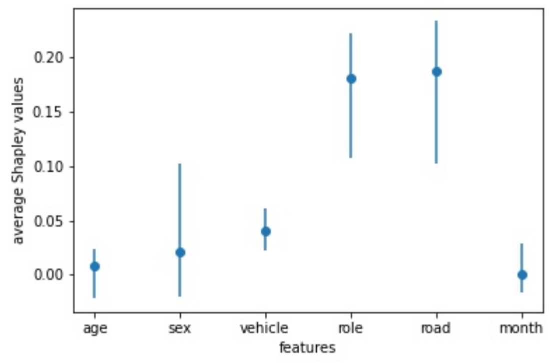

Fig. 1. Plot of the Shapley values with relative max and min values for the algorithm that uses the first v function.

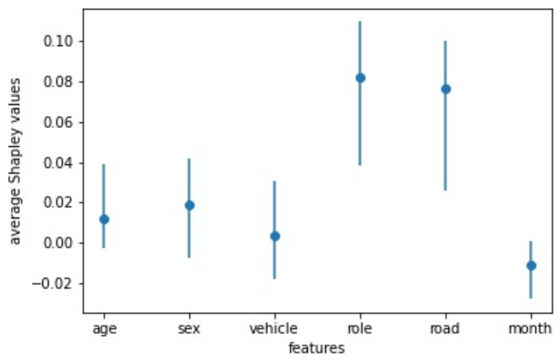

Fig. 2. Plot of the Shapley values with relative max and min values for the algorithm that uses the second v function.

time we had to retrain the model f on the new dataset. As we can see from these figures the evaluated Shapley values are consistent both relatively with each other and with our intuition: most importance has been given to the individual's role and the road type whereas the month has little to no influence over our predictions. The vertical bars represent the maximal and minimal value taken for each feature across the 10 predictions. At first we could be surprised by the Shapley value relative to the vehicle being so low, however, we have to take into consideration that the dataset has been generated uniformly and hence the feature *car* will appear far too rarely for it to be a reliable indicator of whether an accident has happened or not.

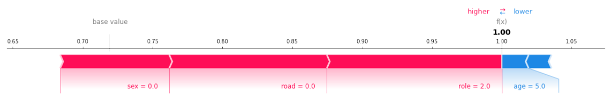

Fig. 3. Explanation of a specific output for an observation of D_{test}. The observation reported an incident involving an individual defined by the vector $[5, 0, 1, 2, 0, 2]$. So someone at least 65 years old, male, on a bike, as the driver, on an urban road on March.

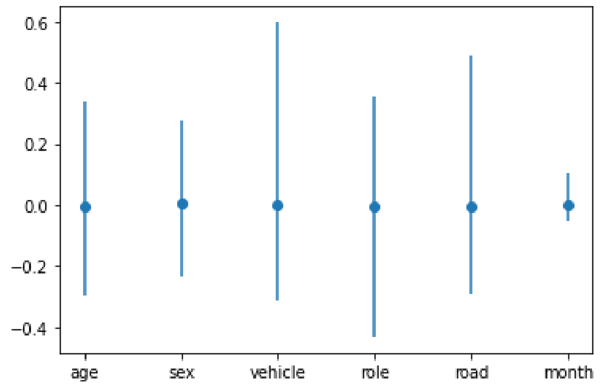

Fig. 4. Plot of the Shapley values with relative max and min values for the algorithm that uses the second v function.

4.2.2. *Local explanation*

Next we have fixed an observation from our dataset and computed the Shapley values of such observation according to Algorithm3. As we can see from Figure3 the Shapley values sum up to a local explanation, i.e. their sum explains the difference between the model output and the expected output. Moreover we have evaluated the Shapley values for all observations in the test part of the dataset, in order to gain a global understanding of the feature's importance. However this search does not yield results like the ones depicted in Figures 1 and 2, search that we have carried out by plotting the average of these 'local' Shapley values in Figure 4. Finally, in Figure5, we have plotted the (sorted) error between the actual value $f(x^*)$ and the explanation $E[f(x)] + \sum_{i=1}^{d} \Phi_i(x^*)$ in order to showcase how good our explanation is. These figures allow us to

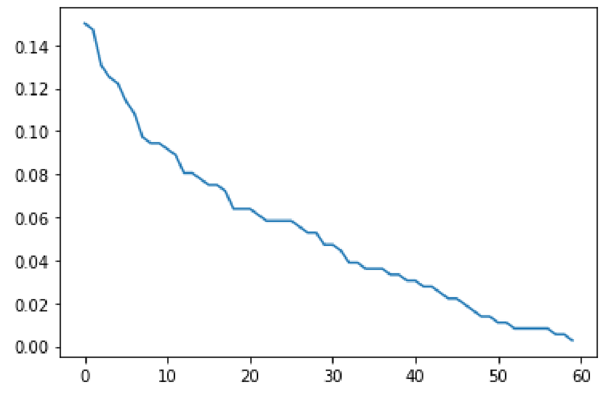

Fig. 5. Sorted graph of the error of explanation $|(E[f(x)] + \sum_{i=1}^{d} \Phi_i(x^*)) - f(x^*)|$. Notice how the worst error has been little more than 0.14 which is quite small when compared to the output 1.

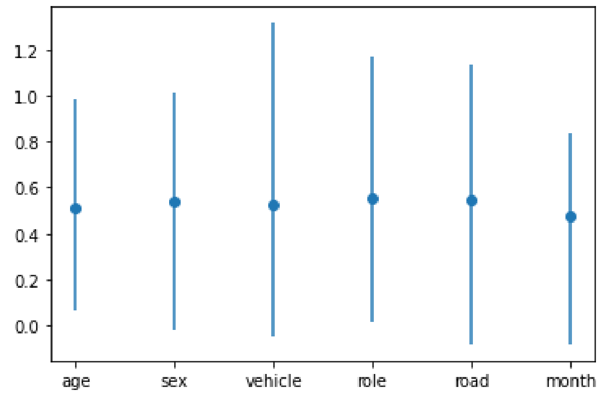

Fig. 6. Plot of the Shapley values with relative max and min values for algorithm 4.

see how, although not being able to extract relevant features from our data, this is locally an excellent explanation model that allows us to efficiently understand what features contributed and in what way to our deviation from the expected value. Finally, let us see the mean Shapley values plot relative to Algorithm4. As we can see from Figure6, also this method doesn't extract global patterns from our features hence, as it is also more computationally expensive, we will not use it further.

4.3. Taking advantage of dimensionality reduction

Finally we have used the data obtained by applying Algorithms 1 and 2 about which features were the most relevant in order to train a new model only over those features. In this way not only did we manage to significantly reduce the overall computation time but we also obtained a more pecise model. As we can see from Figure7 the models that were trained only on the two and three most influential features actually outperformed the model that was trained with all features present. The height of each bar corresponds to the number of misclassifications (the incident happened but it was not predicted, or the incident wasn't predicted but actually happened). Each bar corresponds to a machine's output where each machine was trained as follows:

f1,v1 Is the machine trained only on the most influential feature according to the first worth function.

f2,v1 Is the machine trained on the two most influential features according to the first worth function.

f3,v1 Is the machine trained on the three most influential features according to the first worth function.

f3,-v1 Is the machine trained on the three least influential features according to the first worth function.

f1,v2 Is the machine trained only on the most influential feature according to the second worth function.

f2,v2 Is the machine trained on the two most influential features according to the second worth function.

f3,v2 Is the machine trained on the three most influential features according to the second worth function.

f3,-v2 Is the machine trained on the three least influential features according to the second worth function.

f Is the machine trained on the full dataset.

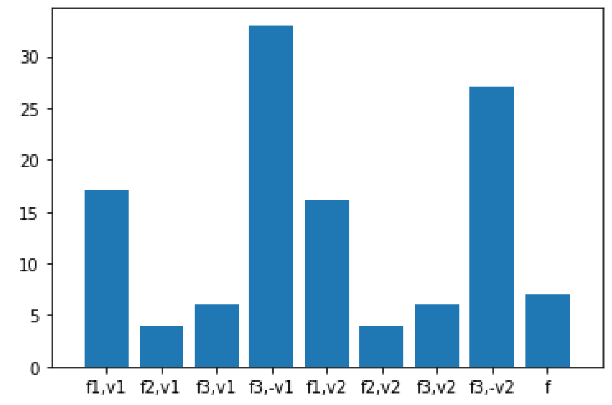

Fig. 7. Histogram representing the number of misclassifications for the differently trained models.

5. CONCLUSIONS

In this paper we have dealt with the tasks of dimensionality reduction through feature extraction and machine learning's explainability with the use of the Game Theory concept of Shapley value. Firstly we developed different algorithms to evaluate the Shapley values. Then we compared their performances and capabilities. Finally we took advantage of the found Shapley values in order to train another model which is faster and more accurate. The programs and concepts present in this paper could actually be expanded in order to broadly apply these tecniques.

References

[1] Kjersti Aas, Martin Jullum, and Anders Løland. *Explaining individual predictions when features are dependent: More accurate approximations to Shapley values*. 2020. arXiv: 1903.10464 [stat.ML].

[2] Haneen Alsuradi, Wanjoo Park, and Mohamad Eid. "Explainable Classification of EEG Data for an Active Touch Task Using Shapley Values". In: *HCI International 2020 – Late Breaking Papers*. Ed. by Constantine Stephanidis et al. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Germany: Springer Science and Business Media Deutschland GmbH, 2020, pp. 406–416. DOI: 10.1007/978-3-030-60117-1_30.

[3] Shaheen S. Fatima, Michael Wooldridge, and Nicholas R. Jennings. "A linear approximation method for the Shapley value". In: *Artificial Intelligence* 172.14 (2008), pp. 1673–1699. DOI: https://doi.org/10.1016/j.artint.2008.05.003.

[4] I.Stat. *Incidenti stradali*. URL: https://www.istat.it/it/files/2020/07/Incidenti-stradali-in-Italia-Anno-2019-aggiornamento27ottobre2020.pdf.

[5] I.Stat. *Incidenti stradali con lesioni alle persone*. URL: http://dati.istat.it/Index.aspx?DataSetCode=DCIS_INCIDENTISTR1.

[6] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *CoRR* abs/1705.07874 (2017). URL: http://arxiv.org/abs/1705.07874.

[7] Benedek Rozemberczki and Rik Sarkar. "The Shapley Value of Classifiers in Ensemble Games". In: *CoRR* abs/2101.02153 (2021). arXiv: 2101.02153. URL: https://arxiv.org/abs/2101.02153.

[8] L. S. Shapley. "17. A Value for n-Person Games". In: *Contributions to the Theory of Games (AM-28), Volume II*. Ed. by Harold William Kuhn and Albert William Tucker. Princeton University Press, 2016, pp. 307–318. DOI: doi:10.1515/9781400881970-018. URL: https://doi.org/10.1515/9781400881970-018.

[9] Ministero delle infrastrutture e della mobilità sostenibili. *Incidenze % per categoria di veicolo*. URL: http://dati.mit.gov.it/catalog/dataset/incidentalita-stradale-anno-2018/resource/d52bc422-fece-48f3-8ca9-31e5a6e8cd6ff.