Understanding Emotion Classification

Through Shapley Values

Noah Lehmann
noah.lehmann@hof-university.de
University of Applied Sciences
Hof, Germany

ABSTRACT

In a time where Artificial Intelligences (AIs) offer more and more applications it is vital to have full control and understanding of how the models come to their decisions. Take for instance the field of medicine, where AIs can be used to determine whether a picture of some human fiber does indeed carry malicious deceases. In such cases the user - in the example given being a doctor - must understand how the AI came to its conclusion, be it just to verify the correctness of the outcome. As the Models used in AIs get more complex and transform to black box models, it gets a lot harder to understand the inside of such a model. This is where Shapley-Values can help developers and users to understand the importance of features affecting a model in its decision process. Shapley-Values come from a game theory background and test the contribution of input features to the outcome of a model by comparing the feature's contribution in comparison to the contribution of all possible combinations of features and their weighed importances. By simply testing all combinations of input to the model, the application of Shapley Values is model-agnostic, their only downside being the computation time in complex scenarios.

KEYWORDS

Explainable AI, Shapley Values, Emotion Analysis, Natural Language Processing

1 INTRODUCTION

Machine Learning (ML) has come a long way in the last few years and starts to unfold possibilities that nobody could have thought of just recently. Distributed learning for instance makes it possible to train highly sophisticated models while still preserving the data's owners privacy rights. On the other hand, AIs have come far enough to beat human professionals in all kinds of quests, making them more powerful - even if only highly specialised.

The challenge now is to find ways to actually leverage use from these powerful mechanisms, preferably in providing humans assistance in situations where it might not have been possible without AIs. One way is to use this computed assistance in situations where humans tend to isolate themselves, especially when they start to develop depressive tendencies. It is not just the isolation, in some cases the affected people don't recognize their tendencies themselves - thus not seeing the need for help from a professional.

In these cases, *emotion analysis* may come in handy, as the affected persons texts, posts on social media or general communication can be analysed and later categorised, thus recognizing patterns early and then acting in a defined way to take the first steps in helping the user to overcome his or her problems.

The emotion analysis described here comes with multiple caveats, mainly in the concern of data privacy and regulations concerning the analysis of the written data and in the way the models recognize patterns and thus decide on whether they have spotted a pattern of depression.

As this study paper is only trying to show the potential of the idea, the data privacy regulations and all other aspects to be considered when handling and analysing user data will not be discussed. The consideration of how the models make their decisions and what makes them decide as they do will be a main part of the analysis though and will be discussed thoroughly later in the paper.

1.1 Emotion Analysis

In order to see the big picture in the explanation of the models decision, it is important to have a test scenario to which one can refer. As explained, AIs tend to take more and more important tasks, thus making it more important to understand their decision flows.

One sensitive topic is emotion analysis, where a user could potentially be classified as depressive in how they write and speak. Emotion analysis could gather the users outputs in multiple formats, analyse them in bulks and recognize major changes in their behaviour. This can be used in different ways, on one hand recognizing texts that should be banned from social media, such as hate speech and similar types of texts, on the other hand this can, as described before, also recognize depressive tendencies, thus providing help in early stages.

Chapter 3 will discuss such an approach to *sadness*-detection, alongside a short summary of an implementation of an emotion analysis application and the explainability of this application through Shapley Values.

1.2 Explainable AI

As mentioned before AIs get more and more complex. Especially looking at models like Neural Networks (NNs) becoming more complex, it is vital to be able to understand the decision-making process of these models. The concept of explaining a models way of working is called Explainable AI (XAI). The key idea of this concept can be described as followed:

> It is impossible to trust a machine learning model without understanding how and why it makes its decisions and whether these decisions are justified[4].

As a result of AI having to solve more and more complex and important tasks, humans need to be able to find errors or potential problems in their training and functionality. The following chapters will discuss an attempt to XAI through the calculation Shapley

Values, followed by an application which attempts to understand emotion analysis through Shapley Values.

2 SHAPLEY VALUES

As aforementioned, the complexity and importance of decisions AIs make shapes the need for a way to understand how the decisions were actually made. The concept of explaining a models way of working is called XAI - as discussed in chapter 1.2. One technique to achieve this in a sufficient manner is the calculation of *Shapley Values*. The following chapter will describe the concepts of Shapley Values and show its usage in an example.

2.1 Background

In order to grasp the idea of Shapley Values fully, it is helpful to look at the background of Shapley Values. Shapley Values have their origin in the mathematical field of *Game Theory*, which - according to the Cambridge Dictionary - is "a mathematical theory about how decisions are made in situations where one person's decision affects another [...][3]". Let's look at this theory from the perspective of an actual game through an example.

Consider a group of players in a tournament forming a team. At the end of the tournament, this team actually wins and will receive a payout as reward for the first place. Looking at the teams efforts, there are generally two ways to distribute the payout fairly.

(1) **Equal Payout:**
In a team which can only win, if all players are present and participating, one can consider the contribution or importance of each player as equal, thus making an equal payout the only fair option, as the team would not have been able to compete without any of its members.

(2) **Contribution-dependent Payout:**
In a team which will only change its efficiency, if one player does not attend or is handicapped, the payout cannot be calculated as easily. It is now dependent on how each player contributed to the actual outcome and therefore makes the calculation far more complex.

As mentioned, the calculation of an equal payout is trivial, however the calculation of the payout to each player depending on his or her contribution to the outcome is not. The later case is an exact use case for *Shapley Values*. Another real life scenario for the usage of Shapley Values could be the fair distribution of a taxi bill, for which all passengers were driven to a different destination.

2.2 Usage in Emotion Analysis

Shapley Values can be used in understanding a models decision of an emotion classification for a whole text block. In this case each relevant word will be checked for its contribution to the outcome as a whole. Given the sentence *I am very sad today*, a well-trained model would come to the conclusion, that this text indeed has a sad motivation. Let's look at the following example in figure 1, where - with a fictional certainty of 90% - a model rated the sentence above as *sad*. Figure 1 shows how the calculation of Shapley Values can divide the contribution from each word to the total of 90% certainty and thus giving the user the possibility to check on the models decision. If it were to interpret contextual non-sad words as contributing largely to the total outcome, a human interpreter

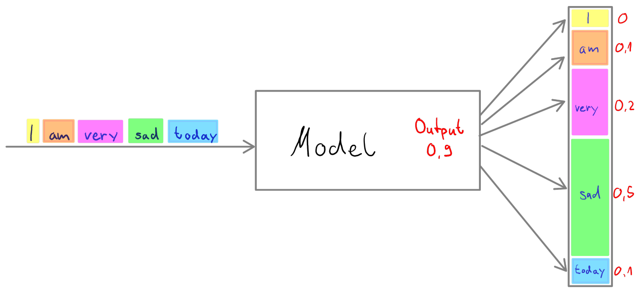

Figure 1: Blackbox feature contribution view

would easily detect an error in the decision, even if the actual outcome was correct.

Notice, how in figure 1 the model is not defined and rather displayed as a blackbox model, meaning that the calculation of Shapley Values happens completely model agnostic. Why exactly this applies will be shown in chapter 2.3.

One more feature of Shapley Values is the calculation and categorisation of an outcome in comparison to the models decision-making in general. If we take a look at the example of emotion analysis again, we could consider a model which tends to classify one in ten texts as *sad*. This gives an initial orientation of how the model tends to work. If we now also take into consideration, how the model reacts to the same word in different sentences, we can see the total benefit of Shapley Values. Figure 2 shows how the calculation can be interpreted with the comparison to the classification of the same words in other sentences. Shapley Values not

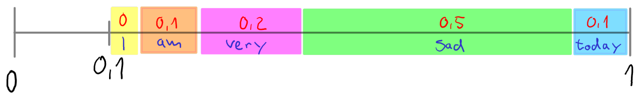

Figure 2: Blackbox feature contribution view

only calculate a feature's - in the case of emotion analysis a word's - contribution to the model's outcome in *one* sentence, they also take the words effects on *other* sentences and classifications into account.

2.3 Mathematical Definition

As Shapley Values have their origin in the mathematical field of *Game Theory*, it is important to understand their mathematical definition, before further examining their applications in XAI and actual implementations[5, 6].

Before looking at the complete formula for the calculation of Shapley Values, some variables need to be defined:

- N = Set of all Attributes $\longrightarrow$ *features or words*
- $n = |N|$
- v = Function $\longrightarrow$ *classifier, black box*

With the most important variables defined, let's look at the definition of Shapley Values:

$$\Phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!}\,(v(S \cup \{i\}) - v(S))$$

Figure 3: Definition of Shapley Values[1]

Φ_i in this case is the contribution of a selected feature i to the outcome of the function v if applied to a sentence with the set of words N. Shapley Values calculate the contribution of each feature to the total outcome, so for an outcome like in figure 1, formula 2.3 must be calculated for all $i \in N$ - meaning for all words in a sentence, referring to the example of emotion analysis.

The rest of the formula generally consists of three parts, the contribution of a feature i to the function v's output when applied to a subset of N excluding i, $S \subset N \setminus \{i\}$, it's weighed importance to other combinations of $S \subset N \setminus \{i\}$ and the sum of all these weighed importances for all possible subsets of N excluding i, $S \subseteq N \setminus \{i\}$, where the empty set $\emptyset$ should not be forgotten.

Looking at these parts separately will make grasping the formula in total easier. Consider the first part of the ones mentioned above, the contribution of the feature i to the outcome of v when applied to a random subset of $N \setminus \{i\}$.

This part states the following: $(v(S \cup \{i\}) - v(S))$. This states, that the function v must be applied to both a subset of $N \setminus \{i\}$, as well as to the same subset including i. The difference in the outcomes of both function calls will then weighed with the importance of the chosen subset $S \subseteq N \setminus \{i\}$. This will be done by looking at the sizes of both S and N and keeping the total count of possible subsets of both in mind.

This is calculated by the following part: $\frac{|S|!\,(n-|S|-1)!}{n!}$. Here the total number of possible subsets of both all elements of S and all other elements in N - a set $X \cup N \setminus \{i\}, S$, are multiplied and then weighed with the total number of all possible subsets of N.

As each weighed contribution of i to any subset S now only has the importance of the size of S, the importances of i need to be compared and summed over all possible variations of $S \subseteq N \setminus \{i\}$. This will calculate the importance of i in comparison to all other attributes - or words - of N.

The summation is declared in the following part: $\sum_{S \subseteq N \setminus \{i\}}$. As mentioned above, the formula only calculates the Shapley Value for one feature i, so for this to be compared to all the other features for an input to a model, the formula must be applied for all features $i \in N$.

2.4 Example

To make the formula more understandable and to reduce complexity to the formula itself, an example can be used to show the calculation of Shapley Values. Consider the use case of a black box classifier, which is trained for recognizing greetings in texts. Given are the Classifier model v and the example Sentence N consisting of the words $\{\text{"}Hello\text{"} = h, \text{"}There\text{"} = t\}$. A classification output of 1 states a recognized greeting with a certainty of 100%, whereas 0 is classified as no greeting with the same certainty.

After running the model on the sentence as a whole and on all isolated words in N, the following is observed:

- $v(\{h\}) = 0.5$
- $v(\{t\}) = 0.25$
- $v(\{h, t\}) = 1$

Considering the need for all possible subsets of N, the following subsets have been identified:

- $\{h\}$
- $\{t\}$
- $\emptyset$

After filling in the needed values, the formula now looks like this:

$$\Phi_i(v) = \sum_{S \subseteq \{h,t\} \setminus \{i\}} \frac{1}{2}\,(v(S \cup \{i\}) - v(S))$$

Like mentioned in chapter 2.3, the Shapley Values' formula now needs to be calculated for all items i in N, being h and t. For the contribution of h the calculations are the following:

$$\Phi_h(v) = \sum_{S \subseteq \{\{t\}, \emptyset\} \subseteq \{h,t\} \setminus \{h\}} \frac{1}{2}\,(v(S \cup \{h\}) - v(S))$$

Likewise for the contribution of t the calculations are the following:

$$\Phi_t(v) = \sum_{S \subseteq \{\{h\}, \emptyset\} \subseteq \{h,t\} \setminus \{t\}} \frac{1}{2}\,(v(S \cup \{t\}) - v(S))$$

This results in the following calculations:

$$\Phi_t(v) = \frac{1}{4} * \frac{1}{2} + (1 - \frac{1}{2}) * \frac{1}{2} = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}$$

$$\Phi_h(v) = \frac{1}{2} * \frac{1}{2} + (1 - \frac{1}{4}) * \frac{1}{2} = \frac{3}{8} + \frac{1}{4} = \frac{5}{8}$$

The Shapley Values for *Hello* and *there* for our black box classifier now are $\frac{5}{8}$ and $\frac{3}{8}$, resulting in an added total of 1, which is the classifiers result for the sentence *Hello there*, as displayed in figure 4.

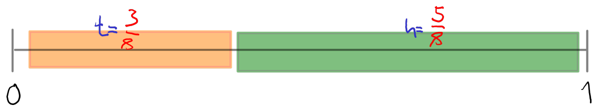

Figure 4: Summation of $\Phi_h(v)$ and $\Phi_t(v)$

2.5 Observations

Additivity. Looking at the example in chapter 2.4 one axiom of Shapley Values gets clear. It is the axiom of *additivity*, which states, that all Shapley Values for a set of attributes add up to the original output of all those attributes combined. This becomes significantly important in the concept described in the beginning of chapter 2.1, where the idea of a distribution by contribution of an outcome is explained.

Model Agnosticism. In principle, the calculation of Shapley Values is model-agnostic. This can clearly be shown in the formula 2.3's definition, where v is only defined as a function, in particular a black box function, meaning that the inner workings of this function are irrelevant and that only the output matters. Especially in the field of AI this marks a benefit of Shapley Values, as they can be used to calculate the feature importance for any kind of model.

Comparability. As explained in paragraph 2.5 - Model Agnosticism, Shapley Values are model-agnostic, thus making them usable for most models. When working with different models, be it in a learning process, decision making process or a development of an AI-product, Shapley Values can be applied to all these models. This helps the user to identify differences in the decision-making and allows him to compare the models with each other. This feature will be used as an example in chapter 3.

2.6 Limitations

When looking at the definition of Shapley Values, two major limitation strike through.

Complexity. As the calculation has to be done for all features and for all possible subsets, the complexity grows exponentially with the size of the feature set, given the formula for proper subsets of a set N being $2^{|N|} - 1$. In most use cases with large feature sets, a preselection must be done before calculating the Shapley Values for the selected features. An alternative to a preselection of features is a preselection of subsets to be iterated for each feature. Both of the techniques make the use of Shapley Values an approximation of their real outcomes in real use cases.

Another attempt to reduce the computational complexity is to reduce the computation time of the model itself, as it needs to be called twice for each iteration (selected subset) and feature. Approximations like the Local Interpretable Model-Agnostic Explanation (LIME) can be used to reduce the models structural complexity, but again making the outcome an approximation. If other techniques like LIME are used, this also limits the calculation of Shapley Values to the limits defined by the chosen approximation.

Output-Irrelevancy. When applying approximations like LIME as stated above, the subsets usually are not selected evenly, but rather created in a way that serves as closest to other subsets as possible, making the actual values inside the subset a combination of actual values from multiple input examples. Those computed values can cause issues in cases where the combination of such make no sense in the respective context.

Take for instance a model which calculates house price estimations depending on the geographical location. If the *longitude* and *latitude* are now delivered as separate features and chosen as an approximation from different subsets, then the calculation could include housing prices for sites in uninhabitable locations. This again influences the accuracy of the approximated Shapley Values.

2.7 Conclusion

As the limitations above are only applicable when the calculation of the Shapley Values surpass a computational limit, thus requiring an approximation to reduce the complexity, they can be avoided by simple not approximating when possible.

Shapley Values offer a simple way of understanding the feature importances of a model output and thus the model. As the mathematical usage is rather simple, the usage is applicable to a wider user base, even for beginners in the field of XAI. Due to the computational complexity, the explanation of models is somewhat limited to the bulk-processing or continuous processing on the server side. A real time computation and explanation on a users enddevice is hardly doable and therefore not recommendable.

3 SADNESS DETECTION

The following chapter will discuss the scenario set up for the explanation of XAI through Shapley Values. It will prepare all information needed to grasp the concept of emotion analysis and its practical application. An actual implementation will be shown alongside the key components.

3.1 Introduction

The project is written in the programming language *Python 3*. Python provides an easy learning curve for beginners, as it can be used to quickly implement prototypes and offers lots of libraries, of which some will be shown later in this chapter. The development environment is set up in *colab.research.google.com*, which provides a Python prepared environment and has lots of libraries for the application of AI pre-installed.

Libraries. As an assistance to the programmer, the use of libraries is proposed. In the actual implementation, the following libraries were chosen:

- Natural Language Toolkit (NLTK): The NLTK-Library provides an array of preparation tools for the use with natural language, in the case of this project, text.
- TensorFlow: TensorFlow provides tools for automated workflows to reduce the complexity in the programming and avoid boilerplate code.
- Keras: Keras is a library which facilitates the use of Neural Networks.
- Scikit-Learn: The Scikit-Learn Library is used for the simplified use of common AI tools, like preprocessing steps and models.
- Shapley Additive Exlanations (SHAP): SHAP is a library which provides an implementation of Shapley Values, mostly using approximations as described in chapter 2.3.

Classification. As shortly introduced in this paper, the use case is an emotion classification. When classifying a dataset, the number of possible classes need to be defined beforehand. When using two classes, the classification is called *binary*. As only two classes are needed, fewer data needs to be used for training the model and a high accuracy is achieved a lot simpler. When more than two classes are used, the classification is called a *multiclass classification*, needing more training data, whereas a high accuracy is harder to achieve. As the next paragraph will show, the dataset used for this

Emotion	Text
neutral	There are tons of other paintings that I thin…
sadness	Yet the dog had grown old and less capable , a…
fear	When I get into the tube or the train without …
fear	This last may be a source of considerable dis…
…	

Table 1: Excerpt of the used dataset

project is not enough for a multiclass classification, thus falling back to a binary classification to reduce complexity.

Dataset. The dataset used in this project consists of two attributes, a text and the corresponding emotion that text expresses. It consists of about 11 300 entries and a total of 5 emotions, *anger*, *neutral*, *sadness*, *joy* and *fear*. As mentioned above, the complexity of classifying all emotions is too high, whereas comparing only one emotion, sadness, is possible with the size of the given data. Table 1 shows an excerpt of the dataset used for this project. In order to reduce the complexity of the dataset and convert the classification to binary, all *non-sad* entries were marked as such. After that, a matching number equal to the count of *sad* entries are chosen from the *non-sad* entries, equally representing the original emotions.

3.2 Preprocessing

In order to ease the computation of text, some preprocessing must be done. As this is not the primary objective of this project, the preprocessing will only be limited to a summary.

Preparation. The first step in the preprocessing is to index the emotions, converting their actual name to an index representing it. As the classification is binary, the indices will be 1 for sad texts and 0 for non-sad texts.

Stopwords. As the complexity of natural language processing rises with long texts containing irrelevant words, these words, also called *stopwords*, need to be removed first. Examples for these words are *I*, *and* or *is*. In this step the unnecessary characters like punctuations or kommas need to be removed as well.

Stemming and Lemmatization. In order to have a clear reference for each word, no matter in what modification they are used, the words need to be stemmed and lemmatized. This means they need to be reduced to their word-stem.

Tokenization. As in the preparation step, the words now need to be indexed to be easily computed.

Padding. In the case of some classifiers, like the Recurrent NN (RNN) used in this project, the sentence lengths need to be aligned in order to be fitted to the algorithms used. This process is called padding. The words can either be filled with blanks to the beginning or end of the sentence or, if the sentences are too long, reduced from the beginning or end of the sentences. This process is not necessary for the other two classifiers used in this project, Naive Bayes and Random Forest.

3.3 Classification

The following classifiers are used in the project:

Naive Bayes. Naive Bayes classifiers are supervised learning methods applying the Bayes-Theorem in combination of the assumption of conditional independence, making it *naive*[7]. The Scikit-Learn library implements multiple Naive Bayes classifiers for an easy usage. The code used in this project can be summed up as followed:

```
from sklearn.naive_bayes \
    import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict(X_test)
```

Random Forest. Random Forest classifiers use an array of decision trees which average their outcomes on samples of the data for a more balanced overall outcome[8]. Again, the Scikit-Learn library offers a simple way to use this classifier, which is shown in the following listing:

```
from sklearn.ensemble \
    import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
```

Recurrent NN. RNNs are NNs which intentionally run multiple time, feeding in an outcome to the next run. They are specifically useful for text processing[2]. Googles TensorFlow library provides easy access to the implementation of NNs and is therefore used in the project as follows:

```
from tensorflow.keras.models \
    import Sequential
model = Sequential()
model.add(Embedding(...))
model.add(Bidirectional(...))
model.add(Dense(...))
model.add(Dense(...))
model.fit(...)
```

3.4 Explanation

Looking at the classifiers above, they work quite differently compared to each other. Even though their output is in the same format, configuring and interpreting their outcomes is not as easy. This is where Shapley Values can help to simplify the explanation of the outcomes through checking each words contribution to the classifiers decision of whether the input sentence was *sad* or *non-sad*.

The SHAP Python library offers an easy way to use already implemented forms of Shapley Value calculations with built-in optimizers to reduce computing complexities. As the definition of Shapley Values and their possible optimizations already have been explained in chapter 2.3, the specific implementation of the used *Kernel-SHAP-Explainer* will not be explained here in detail. In general, it "is a method that uses a special weighed linear regression to compute the importance of each feature”[9]. The usage of SHAP is quite simple and looks like the following:

```
from shap import shap
# x is a sample sentence
shap_rf = KernelExplainer(rf.predict, x)\
    .shap_values(x)
shap_nb = KernelExplainer(nb.predict, x)\
    .shap_values(x)
shap_rnn = KernelExplainer(model.predict, x)\
    .shap_values(x)
```

The following example text will be used to compare the outputs of the three classifiers, which all classified the sentence as *sad*.

Yet the dog had grown old and less capable, and one day the gillie had come and explained with great sorrow that the dog had suffered a stroke, and must be put down.

If the Shapley Values are now to be plotted as bar graphs, to make them more interpretable, the following code can be executed:

```
from shap import shap
shap.initjs()
summary_plot(shap_nb[...],   ...)
summary_plot(shap_rf[...],   ...)
summary_plot(shap_rnn[...],  ...)
```

The graphs now show the following: Even though the Naive

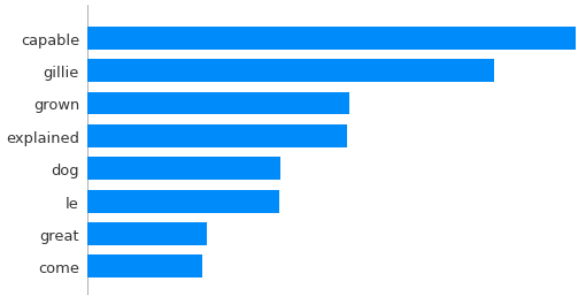

Figure 5: Top 8 Shapley Values for the Naive Bayes classifier

Bayes classifier correctly identified the sentence as *sad*, the outcome seems to be rather incidental, as the words shown as highly influential do not have any sad tendencies. The Random Forest classifier identified more words as correctly sad, which makes it generally more accurate, but still not trustable. The RNN offers the best result in comparing the Shapley Values through all the used classifiers. It correctly identified the sentence as *sad* as well as interpreting the correct words as highly influential.

3.5 Summary

As the focus was not set on the correct setup of the classifiers, slight misinterpretations can be seen in all results shown by the calculated Shapley Values. Instead of reducing the works’ outcome, this actually enhances the view on the benefits of Shapley Values. Comparing the three classifiers clearly shows, that the technique is model-agnostic. Also, if one looks at the outcomes, it is clear that the explainability in these classifications shows that the decisions

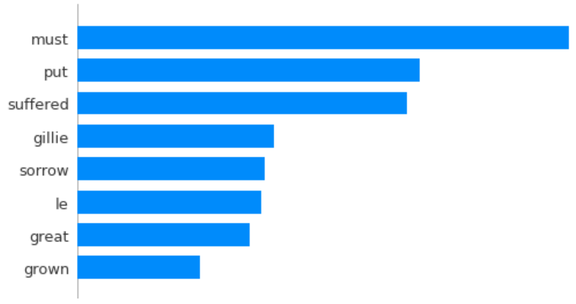

Figure 6: Top 8 Shapley Values for the Random Forest classifier

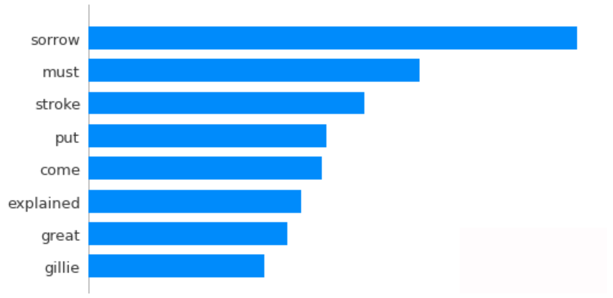

Figure 7: Top 8 Shapley Values for the RNN

themselves where not flawless, which would have been less obvious if only the models outputs had been taken into consideration, as they all classified the example sentence as correctly *sad*.

In summary, Shapley Values offer an easy introduction to the field of XAI, as their calculation is rather simple and they can be used in a wide range of use-cases, given by their mode-agnosticism.

REFERENCES

[1] Marko Cotra. 2019. *Making Sense of Shapley Values.* https://towardsdatascience.com/making-sense-of-shapley-values-dc67a8e4c5e8 [Online; accessed 21.12.2021].
[2] Google Developers. 2022. *Machine Learning Glossary - Google Developers.* https://developers.google.com/machine-learning/glossary/#recurrent_neural_network [Online; accessed 10.02.2022].
[3] Cambridge Dictionary. 2022. *Game Theory, meaning in the Cambridge English Dictionary.* https://dictionary.cambridge.org/dictionary/english/game-theory [Online; accessed 07.02.2022].
[4] Divya Gopinath. 2021. *Picking an explainability technique.* https://towardsdatascience.com/picking-an-explainability-technique-48e807d687b9 [Online; accessed 29.11.2021].
[5] Divya Gopinath. 2021. *The Shapley Values for ML Models.* https://towardsdatascience.com/the-shapley-value-for-ml-models-f1100bff78d1 [Online; accessed 24.11.2021].
[6] Matthew O. Jackson, Kevin Leyton-Brown, and Yoav Stoham. 2013. *GTO-7-03: The Shapley Value.* https://www.youtube.com/watch?v=qcLZMYPdpH4 [Online; accessed 21.12.2021].
[7] Scikit learn Developers. 2022. *Naive Bayes - scikit-learn 1.0.2 documentation.* https://scikit-learn.org/stable/modules/naive_bayes.html [Online; accessed 10.02.2022].
[8] Scikit learn Developers. 2022. *sklearn.ensemble.RandomForestClassifier - scikit-learn 1.0.2 documentation.* https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Online; accessed 10.02.2022].
[9] Scott Lundberg. 2022. *SHAP Kernel Explainer - SHAP latest documentation.* https://shap-lrjball.readthedocs.io/en/latest/generated/shap.KernelExplainer.html [Online; accessed 10.02.2022].