



MASTER RESEARCH INTERNSHIP



INTERNSHIP REPORT

When are anchors based explanations not a good idea?

Domain: Interpretability in Machine Learning

Author:
Julien DELAUNAY

Supervisors:
Luis GALÁRRAGA
Christine LARGOUËT
LACODAM-IRISA



Abstract:

Nowadays almost every domain of activity relies on complex algorithms and models, such as deep neural networks or random forests for multiple tasks suchlike prediction and decision-making. These methods are, however, not interpretable for users. Thus, the European Union has introduced a law on the right for explanations. This law, named GDPR (*General Data Protection Regulation*), permits individuals to ask for an explanation of the decision made by an algorithm on his personal data. To solve this problem, techniques such as Lime [18], SHAP [14] or Anchors [19] have been implemented and allow to interpret black box models by making explanations on how the model works. In this report, we mainly emphasis on Anchors, that approximates complex models locally based on decision rules. We accent on several limits of Anchors and propose solutions to resolve them. The report is structured as following: firstly, we introduce the notion of interpretability for machine learning models thereafter, we survey the most relevant endeavors in interpretable machine learning and classify them into two majors classes: linear attribution and rule-based explanations. Afterwards, we present Anchors more in details and his way of work on tabular and textual data before introducing our contributions and results on Anchors.

Contents

1	Introduction	1
2	Preliminaries	2
2.1	Black-box vs. White-box models	2
2.2	Explainability vs. Interpretability	2
2.3	Instances and Features	2
2.4	Global vs. Local Interpretability	2
2.5	Model-Agnostic vs. Model-Dependent Explanation Methods	3
2.6	Post-Hoc Interpretability	3
2.7	Fidelity and Accuracy	3
2.8	Interpretability/Accuracy Trade-off	3
3	State of Art	4
3.1	Traditionally Interpretable Methods	4
3.1.1	Linear Models	4
3.1.2	Rule-Based Approaches	4
3.2	Linear Explanations	5
3.2.1	Lime	6
3.2.2	Local interpretation methods using the domain of the feature space	7
3.2.3	SHAP	8
3.2.4	DeepLIFT	10
3.3	Rule-Based Explanations	12
3.3.1	Anchors	12
3.3.2	Lore	13
3.3.3	Post Hoc Interpretability for Recommendation Systems	14
3.4	Overview of the different methods	15

4	How to assess the quality of an anchors?	16
4.1	Anchors further in details	16
4.2	Evaluation Measures Applied to Tabular Data	17
4.2.1	Coverage	18
4.2.2	Precision	18
4.2.3	F1 Measure	20
4.3	Evaluation Measures Applied to Text Data	20
4.3.1	Coverage	21
4.3.2	Precision	22
4.3.3	F1 Measure	22
5	Contributions	22
5.1	Impact of Discretization on Anchors Tabular	22
5.1.1	K-means	23
5.1.2	MDLP	25
5.2	Improve Anchors Text Quality with Pertinent Negatives	25
6	Experimentations	27
6.1	Datasets	27
6.2	Black-box Models	28
6.3	Details Implementations	29
6.4	Results	29
6.4.1	Impact of Discretization on Tabular Data	29
6.4.2	Textual Results	33
7	Future Work	35
8	Conclusion	36

1 Introduction

As machine learning models gain ground in critical areas such as medicine, criminal justice systems or financial markets, the inability of humans to understand these models has become problematic. It raises issues of trust about their functional behavior. Consequently, interpretability in machine learning and data mining has become a very active research area. Politicians, researchers and journalists have expressed their concerns about decisions produced automatically by algorithms and require interpretations for both ethical and practical reasons. This point is particularly important in safety critical applications such as medicine or self-driving cars where a wrong decision can lead to human losses. There is a risk that a model makes a wrong decision based on systematic bias in the training data. For example, the survey provided in [10] shows the case of a military program that trained a model to classify friendly tanks from enemy tanks. The classifier shows high performance to recognize them on the training and testing datasets but had very poor accuracy when used in real conditions. US army finally discovered that friendly photos were taken on sunny backgrounds, while enemy photos were taken on cloudy days. Similarly, [18] shows a bad classifier trained to classify husky and wolves. This model made predictions based on the presence or absence of snow in the background. Those insights would have not been possible without the current advances in interpretable machine learning.

The European Union issued a new law, called GDPR (*General Data Protection Regulation*) in April 2016 that has been applied in May 2018. This law is considered as the reference text in terms of protection of personal data. The main objectives of the GDPR are to increase both the protection of data subjects when processing their personal data and the responsibility of those involved in such a processing. This statement requires also a **right to explanation**, and moreover, that algorithmic decisions should be contestable. As Lipton declares in [13], recidivism prediction algorithms such as COMPAS¹ are already used to determine who to detain and who to release from prison. This problem raises ethical concerns such as *how can we be sure that predictions do not discriminate on the basis of race?* As illustrated in [10] or [4], models may incur ethical issues. During the 1970s and 1980s in a hospital medical school from London, a computer program for job applicants was deployed. This program used information without any reference to ethnicity from applicants form. Nevertheless, it was found to unfairly discriminate against women and ethnic minorities by deducting these information from name and place of birth. In these conditions, how a company or a government can trust an algorithm without understanding the behavior of their machine-learning components? The answer for fairness, relies on interpretable models. However, as interpretability becomes an important research area, there are plenty of interpretable models. In this report we present some of them and highlight some of their issues. During the internship, we mostly focus on Anchors [19], an explanation method that approximates complex models locally based on decision rules and the discovering of several limits. Throughout this report, we explain how our research leads to the discovery of these limitations and the resulting improvements we propose to obtain better anchors.

This report is structured as follows: In the first part, we define preliminary notions and introduce the domain of interpretability and its challenges. Thereafter, we present explanation methods for supervised learning starting with linear attributions methods, then followed by rule-based explanations. Subsequently, we go further in the presentation of Anchors and specifically on its use with tabular and textual data before introducing our contributions on Anchors for tabular and tex-

¹Correctional Offender Management Profiling for Alternative Sanctions

tual data. The first contribution is a study of the impact of the discretization method on tabular data, whereas the second contribution is the introduction of a novel explanation method based on Pertinent Negatives for textual data.

2 Preliminaries

2.1 Black-box vs. White-box models

In his work presenting interpretability, Lipton [13] claims that a machine learning model is transparent (a *white box*) if a person can understand the entire model at once. It means that a human should be able to make a decision in a *reasonable* amount of time with the input, the output data and the parameters that define the model. White-box models are used to generate explanations for black boxes. Guidotti *et al.* in their survey [10], describe a black box predictor as an obscure model whose internals are either unknown or known but uninterpretable to humans. In this work, we consider models such as neural networks, random forests, support vectors machines, logistic regression, multinomial naive Bayes as black box models. Some models such as decision trees and linear models may be considered as white box models whereas in some situations, they become too complex and are considered as black box model for the reasons describe upon.

2.2 Explainability vs. Interpretability

Distinguishing between explainability and interpretability is a major question in machine learning. Interpretability defines the ability to predict what is going to happen, given a change in input or algorithmic parameters. It is being able to understand how an algorithm will perform with a new instance. Explainability, on the other hand, is the degree to which the internal mechanics of a machine learning system can be described in simple human terms. There is a slight distinction with interpretability: interpretability is the ability to discern the mechanics without necessarily knowing why. Explainability means being able to describe exactly what is occurring. To the best of our knowledge, the distinction is not well established. Hence, for the rest of this report we apply this following intuition: we consider interpretability as the property of a model to be self explained. For instance, to predict for any instances the illness of a patient (*i.e.*: *A simple linear or rule based model*) whereas explainability corresponds to the model application on a use case.

2.3 Instances and Features

An instance corresponds to an individual from a dataset. This instance is defined by several features that are its information. In a medical use case, the instance is a patient while the features are his symptoms (*e.g.*, *age*, *cough*, *heartbeat frequency*...). In the domain of computer vision, an instance represents an image while the features are the values of the color channels of each pixel.

2.4 Global vs. Local Interpretability

We consider a model as *globally* interpretable if we are able to understand the whole logic of the model. It means that we are able to follow the entire reasoning leading to all different possible outcomes. This is different from *local* interpretability where it is possible to understand the behavior of the decision for a given instance. To resume, a model is locally interpretable if we are able to understand the prediction of the model for a selected instance (*i.e.*, *we understand why a*

model predicts heart problem for an old man that has irregular heartbeat frequency.) Some local explanations models, such as Lime [18] or Anchors [19] can be used to characterised the global behavior of a complex model based on a selection of local explanations.

2.5 Model-Agnostic vs. Model-Dependent Explanation Methods

Most explanation methods introduced in this report are *model-agnostic*, meaning that they can be used to explain all kind of black-box models (*i.e.* *Random Forest, SVM, neural networks...*). Such interpretability models are flexible in the sense that they can be applied to any machine learning model without modifications. However, *dependent models* are designed to explain only one kind of machine learning method. DeepLIFT [20] aims at explaining deep neural network while the method presented in [17] is computed in order to explain the complex behavior of latent matrix factorisation models for recommendation. These explanations concentrate in clarifying only a specific model, thus they exploit knowledge about the architecture of this complex model to obtain better results or less computation time.

2.6 Post-Hoc Interpretability

All the explanation methods presented in this report are considered as *post-hoc*, meaning that they explain a black-box model through a surrogate model or artifact (*e.g., a figure*) on the answers of the black box. There are different processes for taking decisions and explaining them. Post-hoc methods are necessary when there is no way to solve the task in an interpretable way without sacrificing accuracy.

2.7 Fidelity and Accuracy

An explanation is *accurate* if it mimics the black-box model accurately for unseen instances while the *fidelity* is the capacity of a model to imitate a black box predictor. Intuitively, an accurate explanation method is able to predict right classes for unseen instance while the fidelity means that it classifies instances according to the complex model prediction. The accuracy of a model can be quantify using various evaluation metrics like the accuracy score or the F1-score (when applicable). The accuracy measures the number of correct positive results divided by the number of all positive results returned by the classifier.

Producing an accurate interpretable model is the most common purpose among works in the literature. On the other hand, fidelity measures the ability to mimic the behavior of a black box. Fidelity uses similar evaluation metrics but with respect to the outcome of the black box.

2.8 Interpretability/Accuracy Trade-off

Complex methods such as deep neural networks or SVM are among the most accurate models. Their precision is due to ability to capture non-linear relationships between the different variables used for prediction. Whereas models such as logistic regression and decision trees are more interpretable, they are less accurate due to their simple logic. This problem leads to one of the main issues interpretability faces: *How can we gain interpretability without sacrificing accuracy?*

3 State of Art

3.1 Traditionally Interpretable Methods

In this section, we introduce linear models, rule-based and regression trees approaches and show why most of the researchers on interpretability agreed to consider them as interpretable. Miller in his work on social sciences [15] specifies that human has 5 ± 2 cognitive chunks and that explanations based on more than this number of features can be considered as complex. To illustrate our ideas, we used a running example where the goal is to predict a patient's heart illness.

3.1.1 Linear Models

Linear models are certainly the most basic of all machine learning models. Linear models seek to express a random variable as a function of explanatory variables in the form of a linear operation. This model is interpretable since features are associated with a sign and a weight. If an attribute has a higher weight value than another, it means that it has a higher influence on the prediction of the model than the other attributes. A linear model may predict risk of heart attack y with interpretable coefficients of correlation θ_i between the age of the patient x_1 and his heartbeat frequency x_2 as in this formula:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \quad (1)$$

3.1.2 Rule-Based Approaches

Decision/Regression trees are based on nested *if-else* conditions. They create a tree in which nodes split instances from the dataset into two or more homogeneous sets based on the most significant feature. In a decision/regression tree, the importance of the feature that divides the population is represented by the height of the node in the tree. As shown in figure 1, depending on the individual *age*, his *heartbeat* frequency has a bigger impact to determine his risk of heart attack than the *cough* or conversely. This method is interpretable since from a target instance, we follow the path from this instance to the root and it gives us how the model works.

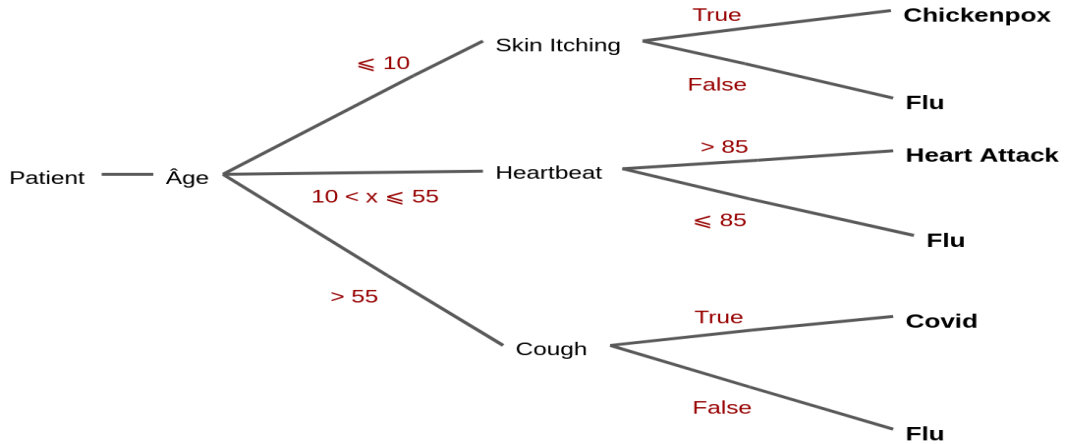


Figure 1: A decision tree predicting disease of an individual based on his age and different features

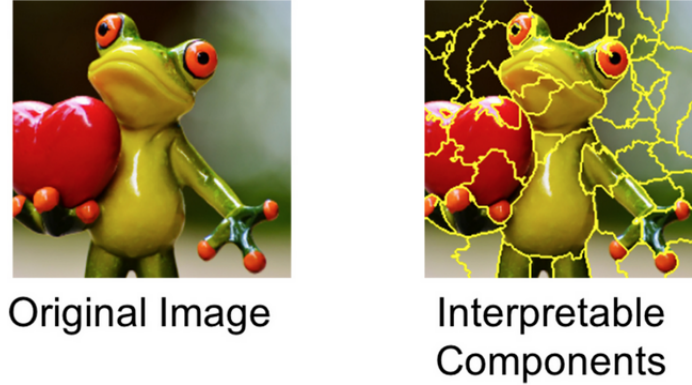


Figure 2: Image from [16] divided into components to test for interpretation by Lime

Rule-based approaches can be viewed as an expert system that produces rules in order to take a decision. A rule contains several conditions, if all these conditions are respected for the instance, then it predicts a label or a numerical value with a certain confidence. In this situation, if all the symptoms correspond to a patient, then the expert system predicts with high confidence an heart attack.

$$R = \{age \geq 55, heartbeats \geq 85, father's\ death = heart\ attack\} \rightarrow \text{heart attack} \quad (2)$$

The interpretation of rules and decision trees is different according to some aspects. Rules have a textual representation while decision trees are generally adopted for their graphical representation. The main difference is that textual representation does not bring information about the most important attributes of a rule. Meanwhile, the hierarchical position of the attributes in a tree provides some hints about their importance. In the case of predicting a patient's heart illness it can be important since the doctor can examine the tree and determine which feature has the greatest impact on the prediction. Thus, he or she can guide questions to determine symptoms more effectively.

3.2 Linear Explanations

In this section, we concentrate on explanation methods based on linear attribution. Recently, two methods Lime [18] and SHAP [14] have stood out among these explanations models. Each method presented in this section approximates a complex model locally with a linear model. In order to be more interpretable, these methods operate on a different representation of the instances based on interpretable features. For example, images are represented as pixels on different color channels, but in an interpretable representation, pixels are aggregated by grouping similar pixels in order to form super pixels as in figure 2. Recent methods on text classification represent a sentence with word embeddings, a complex vector that enables comparison between words and sentences. An interpretable way to characterize a sentence is to indicate the presence or the absence of a word over a vector of words.

3.2.1 Lime

Ribeiro was a Ph.D. student when he first presented Lime [18] in 2016. Ever since, Lime has been a reference in the domain of interpretability in machine learning. Lime is the first method that approximates a complex model locally using linear explanations on instances. Intuitively, Lime generates a set of perturbed instances weighted by their proximity to a given instance. It then learns a linear model to interpret the model behavior by calling the black-box model on this set of generated instances.

Formally, Ribeiro *et al.* [18] define an explanation as a model $g \in G$, where G is a class of potentially interpretable models, such as decision trees, logical rules or linear models. The domain of g is $\{0, 1\}^{d'}$ where g acts over absence or presence of the interpretable features and d' is the size of the vector of interpretable features. If there is no limit in the complexity of a white-box model (*e.g.*, in the depth of a tree), it can become too hard to read and, as suggested by Miller in [15], not interpretable. Hence, $\Omega(g)$ represents a measure of the complexity of the explanation g . In a linear explanation, the complexity may be the number of non-zero weights.

Let the complex model be denoted by $f : \mathbb{R}^d \rightarrow \mathbb{R}$, where $f(x)$ represents the probability that x is associated to a certain value, class or label. We use π_x to denote the proximity measure between an instance z generated and the target instance x as in figure 3. We further use $L(f, g, \pi_x)$ as a measure of how g approximates f in the locality defined by π_x .

In order to provide both interpretability and local fidelity, Lime minimizes the following function:

$$\text{explanation}(x) = \arg \min_{g \in G} L(f, g, \pi_x) + \Omega(g) \quad (3)$$

To be model-agnostic, Lime tries to minimize the locality-aware loss function $L(f, g, \pi_x)$ without making any assumption about f . The interpretable model that explains locally is obtained by drawing samples, weighted by π_x . The samples are obtained from the interpretable representation of x by randomly and uniformly flipping its non-zero elements. Sample instances are denoted $z \in Z^D$ where Z^D represents the space of coalition vector of each sample instance z . Lime calls the model for each instance z to obtain a label for each of them. The weight of each interpretable feature is then calculated using linear regression on the labels obtained from the black box. Lime returns only the K interpretable features that impact the prediction the most. In text classification, the K most important words for the decision are returned while in image classification the K most important superpixels are presented as explanation.

Lime [18] introduces the submodular-pick algorithm. This method aims at explaining the whole model behavior by explaining different instances that are diverse and representative. The intuition behind this algorithm is that users do not need to inspect all instances explanation to trust the model. They need a selection of the *best* instances and an interpretation that explains how the model behaves globally. A good set of instances covers as much data as possible without redundancy. In order to choose these best instances, submodular-pick starts by constructing an explanation matrix for all instances. It then gets different features that work best for most various entities without redundancy. For each component (*i.e.* *superpixels* or *words*) it sums all scores obtained for each entity on this component and measures the global importance of that component in the interpretable space. The main objective of Lime submodular-pick is to return features that have the highest aggregated importance score among the different instances.

A challenge of Lime is the required transformation of new data types such as time series or multimodal data into an interpretable space that is supposedly human interpretable. Another issue

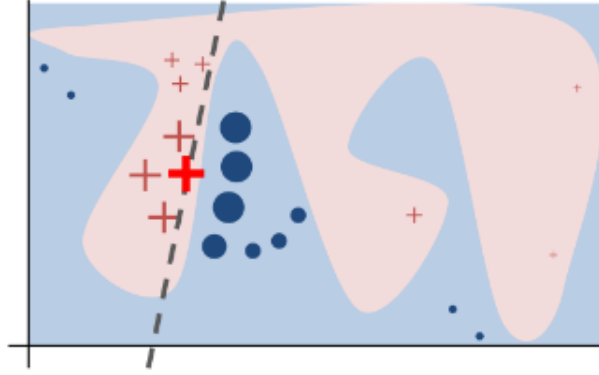


Figure 3: Toy example from [18] to present the intuition for Lime. The *blue/pink* background represents the black-box model’s complex decision function f (unknown to Lime.) Thus, this model cannot be approximated well by a linear model. The bold red cross is the instance being explained x and the other points represent Lime generated sampling method neighborhood. Lime samples instances and gets predictions using f , then weighs them according to the distance function π_x that represents their proximity to the instance being explained (distance is represented here by the size). The dashed line is the learned explanation that is locally (but not globally) faithful.

of Lime is the notion of *proximity*, which is not really defined for the generation of the neighborhood, thus it may lead to instances that are not representative of the problem. Some research has been done to go further and it will be presented in the following.

3.2.2 Local Interpretation Methods Using the Domain of the Feature Space

The work in [3] appears in AIMLAI-XKDD² (2019) and provides a method to improve Lime’s generation of the instances neighborhood. It aims at solving these three questions:

1. How to best define the neighborhood of an instance?
2. How to make the obtained solution robust to small variations on the instance to be explained?
3. How to control the trade-off between the accuracy of the interpretation model and its interpretability?

This method defines the area of instances that are realistic using the α -shape and intersect them with the circular neighbourhood defined by Lime. α -shape is a generalization of the convex hull. Hence, it is based on the concave and convex geometrical ideas. It is looking for convex hull and then filters this result using a circle of radius alpha. Thus, α -shape improves the method to sample some instances from the data set that are closed to the target.

This model presented a new way to generate neighborhood perturbation more robust than in Lime. However, it is important to note that this method needs access to the dataset used to train the black-box model to build the α -shape. Hence, this is an additional assumption that Lime does not make.

²Joint International Workshop on Advances in Interpretable Machine Learning and Artificial Intelligence & eXplainable Knowledge Discovery in Data Mining

3.2.3 SHAP

SHAP [14] is a unified method based on Shapley values. Its purpose is to explain the prediction of an instance x by computing the contribution of each feature to the difference between the prediction and a reference or default value. That contribution corresponds to its Shapley value that we calculate as follows: Let us take one coalition example (*i.e.*: a coalition represents a set of feature values) from figure 4. In order to compute the Shapley value of the feature *cough* for a given instance, SHAP picks random *age* and random *heartbeat frequency* from instances in the database and performs the average contribution of the *cough* feature. Shapley values corresponds to the average marginal contribution of a feature value across all possible coalitions. All possible coalitions have to be evaluated with and without the j^{th} feature to compute the exact Shapley value of this j^{th} feature. To run a model without a feature, it replaces the values of features that are not in a coalition with random values from the dataset. For datasets with many features, computing the exact solution becomes problematic as the number of possible coalitions increases exponentially as more features are added.

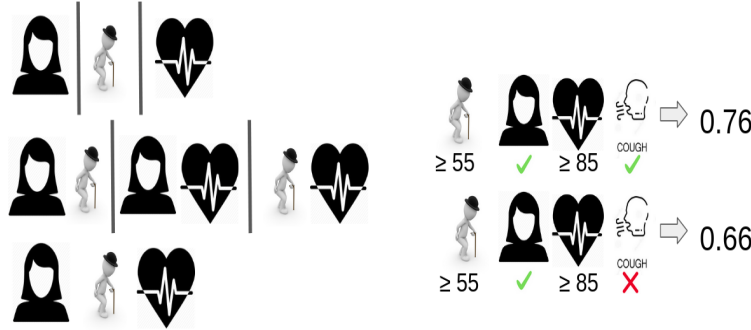


Figure 4: On the left, all 8 coalitions needed for computing the exact Shapley value of the cough feature value. Sex, age and heartbeat frequency are the features that are combined with cough in order to predict the risk of an heart attack. On the right, example of a coalition for a female individual older than 55 with an heartbeat frequency ≥ 85 . SHAP computes for all coalitions the average difference as in this example by computing the difference between coalition with the feature and without: $0.76 - 0.66 = 0.10$.

Shapley values can be misinterpreted as the difference of the predicted value after removing the feature from the model training. Intuitively and in reality, the interpretation of the Shapley value is: *Given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction of an instance is the estimated Shapley value.*

SHAP's novelty lies on providing local explanations in the spirit of Lime [18] or DeepLIFT [20] that satisfy the following three properties based on Shapley values:

1. **Local Accuracy:** Prediction of the interpretable model is faithful with the original model output on the target instance.
2. **Missingness:** If a feature is absent, it has no impact, thus its score is zero.
3. **Consistency:** Given 2 models A and B and a feature x , if the difference in value when A is tested with and without x is larger than for model B , then the Shapley value of x in model A should be larger than the Shapley value of x in model B .

To guarantee these properties, the authors of SHAP has shown that there is only one possible additive feature attribution method. This implies that methods not based on Shapley Values may not conform to local accuracy and/or consistency. In order to achieve these properties, they propose SHAP kernel. The principal difference between Lime and SHAP is the weighting of the instances in the regression model. In Lime, π_x weights generated instances z according to how close they are to the original instance x using a Gaussian kernel. Conversely, SHAP weights z according to the Shapley value estimation. In SHAP an instance that is very close or very far to the original instance has bigger impact on learning of the importance scores as we can see in formula 4. Here, M is the maximum coalition size and $|z'|$ the number of present features in the coalition vector of the instance z' . An instance is close to the original if the coalition vector has many 1's (*i.e. a coalition vector is the vector of presence or absence of features in common*). It means that most of their features values correspond and we can estimate Shapley values of the few different features. The intuition behind this difference is that we learn more on instances that are almost equal or essentially different than half equal instances:

$$\pi_x(z') = \frac{(M - 1)}{\binom{M}{|z'|} |z'| (M - |z'|)} \quad (4)$$

SHAP proposes a model agnostic explainable method based on Lime and Shapley values. It changes the Loss function in Lime to respect these three properties and names it KernelSHAP. In order to make Lime (equation 3) correspond to SHAP kernel formula, KernelSHAP presented these new formulas of the loss function L and regularization term Ω from Lime :

$$\begin{aligned} \Omega(g) &= 0 \\ L(f, g, \pi_{x'}) &= \sum_{z' \in Z} [f(z') - g(z')]^2 \pi_{x'}(z') \end{aligned} \quad (5)$$

We can observe on figure 5 an illustration of SHAP. It explains the predicted cancer probabilities of two women. Each feature value is a force that either increases or decreases the prediction. The average predicted probability of all predictions is 0.066. The first woman has a low predicted risk of 0.06. Red features such as STDs increase risk while blue features such as the age reduce it. The second woman has a high predicted risk of 0.71. It is mostly due to his age and 34 years of smoking.

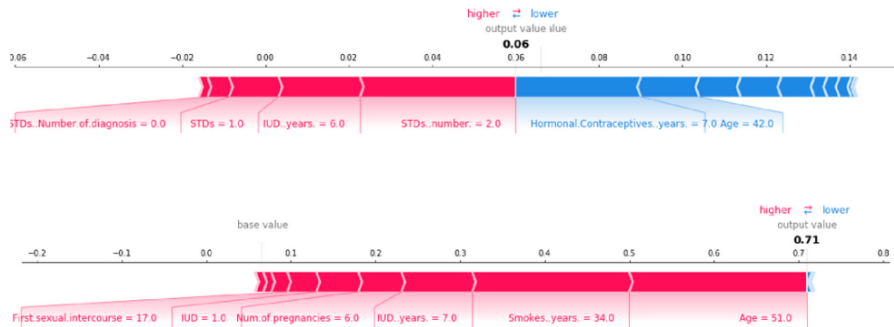


Figure 5: Figure from [16].

SHAP is a unified approach to interpret model predictions, it also presents different approximation methods. We introduced KernelSHAP that is a model agnostic method, but they also proposed methods that are model-dependent. One of them is based on DeepLIFT (that will be presented in the next section) and considers neural networks architecture to improve computational performance.

3.2.4 DeepLIFT

The technique proposed in [20] is a model-dependent explanation method tailored for deep neural networks. It is based on the selection of reference inputs by an expert in order to compute the impact of hidden neurons on the prediction. The reference input is a baseline. Among each intermediate layer, this baseline guides through back-propagation. Hence, it discovers which neurons x_1, x_2, \dots, x_n has the most important impact on the prediction of interest compared to the referent input. On the MNIST digits dataset, all digits are represented by white pixels overlapping a black background. As illustrated in figure 6 a baseline would be a fully black image, thus representing no information about the digits.



Figure 6: Choice of the referent input for the MNIST digits dataset.

Using the neural network, reference input layer leads to a reference output layer. Thus, references for all neurons can be found by choosing a reference input and backpropagating activations through the network. DeepLIFT is based on the *multiplier chain rules* to compute difference Δx between a given input and the referent input and the difference Δt between a given output and the reference output. Multiplier chain rules is similar to partial derivatives in his formulation. It uses the *summation-to-delta* property: $\sum_{i=1}^n C_{\Delta x_i \Delta t} = \Delta t$ where $C_{\Delta x_i \Delta t}$ corresponds to the amount of difference from reference in t that is attributed to or blamed on the difference from reference of x_i . In order to compute the contribution of a target neuron t with difference from reference Δt , DeepLIFT defines a multiplier $m_{\Delta x \Delta t}$ as:

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x} \quad (6)$$

Analogous to how the chain rule for partial derivatives allows to compute the gradient via backpropagation, DeepLIFT uses a chain rule to compute the multipliers for any neuron to a given target neuron efficiently. DeepLIFT introduces three rules for assigning contribution scores; these rules aim to find the contribution of any input to a target output via back-propagation.

Using multipliers it computes the **Linear rule** for dense and convolutions layers. Hence, this rules excludes non linearities. Linear rule gets positive Δy^+ and negative Δy^- components of the linear function $y = b + \sum_i w_i x_i$ with b the bias and w_i weights of the layer for the input x_i :

$$\begin{aligned} \Delta y^+ &= \sum_i 1 \{w_i \Delta x_i > 0\} w_i \Delta x_i \\ \Delta y^- &= \sum_i 1 \{w_i \Delta x_i < 0\} w_i \Delta x_i \end{aligned} \quad (7)$$

The rule that applies to non linear transformations such as ReLU or sigmoid operation is the **Rescale Rule**. It takes the neuron of a non linear transformation of its input and computes the positive and negative impact on the difference from reference:

$$\begin{aligned}\Delta y^+ &= \frac{\Delta y}{\Delta x} \Delta x^+ = C_{\Delta x^+ \Delta y^+} \\ \Delta y^- &= \frac{\Delta y}{\Delta x} \Delta x^- = C_{\Delta x^- \Delta y^-}\end{aligned}\tag{8}$$

The last rule is the **Reveal Cancel Rule**. This rule can be considered as the Shapley values of the positive and negative contribution to the neuron y . It mitigates some of the issues that arise when positive and negative terms cancel each other out. It separates the positive and negative contribution to differentiate the target and the reference (input or output.) Imagine a case where all references values are equals and null. Using *Rescale Rule*, all importance would be assigned either to one of the reference values. In this case, it is not clear that each of the inputs are relevant for the prediction. In the *Reveal Cancel Rule*, it considers the impact of the positive terms without the impact of the negative terms and the impact of the negative terms in the absence of the positive by defining Δy^+ and Δy^- as:

$$\begin{aligned}\Delta y^+ &= \frac{1}{2} (f(x^0 + \Delta x^+) - f(x^0)) + \frac{1}{2} (f(x^0 + \Delta x^- + \Delta x^+) - f(x^0 + \Delta x^-)) \\ \Delta y^- &= \frac{1}{2} (f(x^0 + \Delta x^-) - f(x^0)) + \frac{1}{2} (f(x^0 + \Delta x^+ + \Delta x^-) - f(x^0 + \Delta x^+))\end{aligned}\tag{9}$$

The weights of a linear model might seem intuitive as explanations for the importance of features, yet as Finale claims in his works [7], linear explanations can be counter-intuitive in some cases.

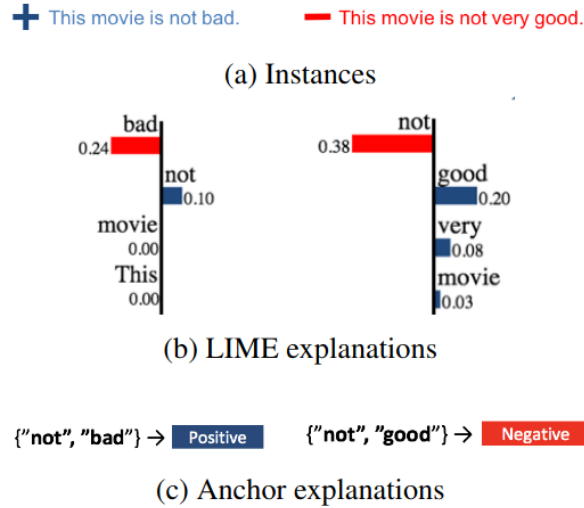


Figure 7: Sentiment predictions based on a LSTM from [19].

For instance, on picture 7, for two sentiment predictions made by an LSTM³, Lime reports a negative impact on the adverb *not* in a sentence including (“*not*” and “*good*”) and a positive impact

³Long-short term memory

for a sentence with (“not” and “bad”). This might not allow to generalize the model. Hence, if a user inspects these 2 examples and want to anticipate the prediction of the LSTM for a new sentence containing the word ”not”, the user can misunderstand previous explanations. Thus, he may try to anticipate in a wrong way based on the wrong example. Consequently, we now present the rule-based methods, which are more easier to generalize since prediction is performed only once all the conditions are fulfilled.

3.3 Rule-Based Explanations

This section focuses on methods that use logical rules to explain locally a complex model. [15] claims that logical rules are considered more easily interpretable by humans than linear models and provide a well-defined notion of explanation coverage. The coverage of a rule is the probability that this rule applies to samples from a set D . In this section we first introduce Anchors [19], and then Lore [9] a new method that tackles some Anchors issues. Thereafter, we present a method to explain matrix factorization methods for recommendation based on association rules [17].

3.3.1 Anchors

Anchors [19] is a novel method introduced by Ribeiro *et al.*(2018) that mines rules to predict a particular class with high probability and coverage. An anchor is a logical rule that predicts the class assigned to a target instance by the black box with a certain confidence. In order to find the best rule to classify an instance, Anchors mines rules with a breadth first search algorithm. There is an exponential number of such logical rules. Thus, searching the best solution is very challenging.

Anchors creates a perturbed distribution D around a target instance x . This method calls the black-box model over sample instances in order to estimate precision and coverage bounds under D . As in Lime [18], Anchors converts instances to an interpretable space on the absence or presence of some conditions or features. Thus, let A denote a rule acting such that $A(x)$ returns 1 if all its interpretable features are true for instance x . Let $\mathcal{D}(\cdot|A)$ describes the conditional distribution that the rule A applies. Let the complex model be denoted by $f : \mathbb{R}^d \rightarrow \mathbb{R}$, where $f(x)$ represents the probability that x is associated to a certain value, class or label. A is an anchor if $A(x)$ is satisfied and is a sufficient condition for the target instance with high probability τ :

$$\mathbb{E}_{\mathcal{D}(z|A)} [I_{f(x)=f(z)}] \geq \tau, A(x) = 1 \quad (10)$$

This model is first based on an algorithm that incrementally constructs the anchor A . It starts from an empty rule (*i.e. one that applies to every instance.*) In each iteration one additional feature predicate a_i is added. By definition, at every iteration the set of candidate rules is $\mathcal{A} = \{A \wedge a_i, A \wedge a_{i+1}, A \wedge a_{i+2}, \dots\}$. The candidate rule from \mathcal{A} with best estimate precision is selected and replaces A . If A satisfies the sufficient threshold probability, then it is selected as the best anchor. This intuition is based on the fact that short rules have higher coverage than larger rules. As mentioned before, there is an exponential number of candidate rules, so pure exploration multi-armed bandit method is employed. In each step, the algorithm selects two distinct rules from \mathcal{A} , the best mean and the highest upper bound. It updates their bounds by getting a sample from each conditional distribution for candidate rules. This sampling process continues until lower bound of the best mean rules is higher than the upper bound of the highest rule.

Anchors has two main issues. First, it maintains a single rule at a time and thus a sub-optimal choice is irreversible. The second main issue is that optimal coverage is not guaranteed.

This method returns the shortest anchor, but not the one with the higher coverage. To solve these problems, Anchors proposes a beam-search algorithm that extends the classic algorithm by maintaining a set of candidate rules while aiming to identify among the possible anchors the one that has the biggest coverage.

As compared to Lime, Anchors has a method to explain a model globally based on submodular pick. The purpose of this method is to give a more complete understanding on how the model works not only for individual instances. To help users, this method gives anchors explanations on diverse set of instances. Submodular pick takes an optimal subset of anchors that covers as many instances as possible to explain the global behavior of the model.

Anchors has several issues and many points require further research. In some cases, explaining an instance with an anchor is very difficult because, when the instance is close to a boundary of the black-box model the anchor can be overly specific. These kind of explanations do not generalize the model enough and users only get that this instance is near a boundary. The second important issue is that Anchors depends on the perturbation distributions of target entities. Finding the good perturbation for various domains remains challenging. To solve these two problems, Lore [9] presented below extends Anchors on tabular data by adding counterfactual rules that indicate for which change in feature value, the prediction varies.

3.3.2 Lore

Lore [9] is a rule-based method suitable for explanation of black boxes on tabular data. Based on a given instance, Lore generates artificial neighborhood instances by means of a genetic algorithm. From the neighborhood instances, and the answers of the black box, Lore learns a decision tree classifier. Local explanations are extracted to obtain a logical rule and a set of counterfactual rules. Counterfactual rules are the minimal number of changes in the feature values of an instance that change the decision of the predictor. It describes *which* features to change and *how* to change them to get a different outcome. Let us consider the example of the following explanation e given by Lore, predicting the illness of a patient on an instance x , where $x = \{(age = 62), (temperature = 38.2), (father's death = natural)\}$

$$\begin{aligned} e = \langle r = \{(age \geq 60), (temperature \geq 38), (father's death = natural)\} \rightarrow flu, \\ \Phi = \{(\{father's death = heart attack\} \rightarrow heart problem), \\ (\{age \leq 5, temperature \leq 36.5\} \rightarrow pneumonia)\} \end{aligned}$$

In this example, r is the decision rule and Φ is a set of counterfactual rules. The decision of the flu prediction changes if the individual, instead of being older than 60 with temperature, is a child and has a low temperature. A second change may occurs if his father died of an *heart attack* instead of a *natural death*. As mentioned earlier, in order to discover decision rules, a decision tree as in figure 1 is learned upon the created perturbed neighborhood. Decision rules are derived from a root-leaf path to the target instance in the decision tree. Counterfactual rules are extracted by following all paths in the decision tree leading to a different outcome from x . Lore does not return all the paths as counterfactual since it is only interested in those having minimum number of split conditions not satisfying x .

Lore improves the generation of perturbed neighborhood from Anchors by implementing an algorithm based on genetic algorithm. The purpose of this step is to identify a set of instances

$z \in Z$ that are close to the target instance x but that is able to reproduce the local decision behavior of the model. This set of instances must contain elements with different outcomes in order to learn a decision tree predictor. Thus, the genetic algorithm creates two sets of instances of the same size that have the same and a different decision value than x . This algorithm minimizes these two fitness functions:

$$\text{fitness}_{=}^x(z) = I_{b(x)=b(z)} + (1 - d(x, z)) - I_{x=z} \quad (11)$$

$$\text{fitness}_{\neq}^x(z) = I_{b(x) \neq b(z)} + (1 - d(x, z)) - I_{x=z} \quad (12)$$

The first fitness function considers generated instances z closed to instance x according to the distance function $d(x, z)$ but not equal to x (*i.e.* term $I_{x=z}$). This function is also researching instances for which the black box model b produces the same output as x (*i.e.* term $I_{b(x)=b(z)}$). The function 12 returns instances closed to x but not equal to x and for which b produces a different output (*i.e.* term $I_{b(x) \neq b(z)}$). This genetic algorithm samples instances based on principles of *crossover* and *mutation* as depicted in figure 8.

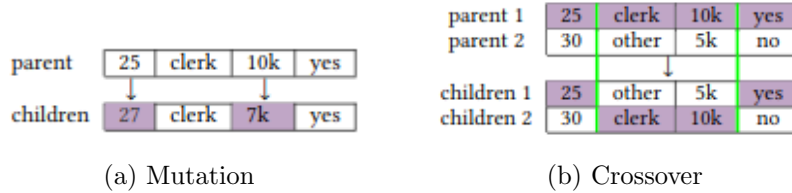


Figure 8: Mutation and Crossover operators. The figures were taken from [9]

Lore introduces a new way of explanations by adding counterfactual rules. Moreover, it relies on a genetic algorithm to generate neighbor instances that could help us in our future research.

3.3.3 Explanation Mining: Post-Hoc Interpretability of Latent Factor Models for Recommendations Systems

Georgina Peake and Jun Wang present in [17] a method to interpret recommendation systems based on association rules. The intuition is that association rules can explain recommendation systems based on latent factor models very intuitively. Hence, based on a matrix factorization that predicts film rated for user, this method produces association rules identifying relationships between films. Experiments are made on TV programs recommendations made via constrained matrix factorization. Matrix factorization initiates with a rating matrix R and then derives a matrix of user factors U and an item factors matrix V . From these two matrices, it computes a rating predictions matrix \hat{R} as in figure 9. This obtained matrix predicts what users will rate based on their similarity to other users.

To explain recommendations for a user, this work introduces post-hoc interpretability explanation method based on association rules. Association rule identifies relationships between items (*i.e.* $Deadpool \Rightarrow Avengers$.) These rules are evaluated by support, confidence and lift. Given a rule $X \Rightarrow Y$, support corresponds to the proportion of transactions containing items X and Y . Here, it means that support corresponds to the proportion of users that have rated both movies X and Y . Confidence is the support of X and Y divided by the number of users who rated X , thus the

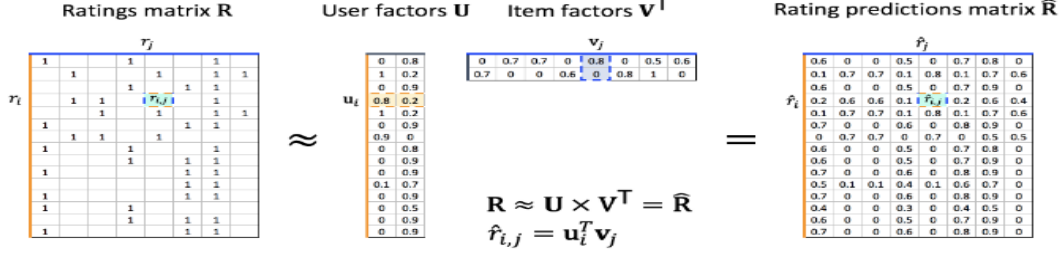


Figure 9: Matrix factorization compressing the user-item matrix R into two lower rank matrices, user U , and item V . The resulting rating predictions matrix \hat{R} is an approximation of the original ratings matrix. Figure taken from [17].

conditional probability. Lift corresponds to the measure of independence between two items. The computation of association rules uses the Apriori algorithm upon the matrix of rating predictions returned by the matrix factorisation model. The model implemented in this article to discover association rules is based on hierarchical clustering algorithm. Clustering algorithms group individuals who share similarities. Thereafter, it mines rules that correspond to the outcomes of the matrix factorisation on those clusters. It delimits different clusters of different size by regrouping them upon their distance to each data point. This method is specific to matrix factorisation prediction models and it is therefore model-dependant.

3.4 Overview of the different methods

In order to summarize all the functionalities of the different methods presented in this report, we provide a table that indicates the weaknesses and strengths of each of them. Since Lime and Anchors are the most common methods, they fulfill most criteria but are less reliable. In this survey, we have nuanced the application of such methods by introducing new methods that offer better performance in certain contexts.

	Lime	SHAP	Anchor	Lore	Association rules	DeepLIFT
High fidelity	😊	😊	😊	😊	😊	😊
High confidence	😊	😊	😊	😊	😊	😊
Interpretable	😊	😊	😊	😊	😊	😊
Computable	😊	😊	😊	😊	😊	😊
Model agnostic	😊	😊	😊	😊	😊	😊
Multiple class	😊	😊	😊	😊	😊	😊
Word representation	😊	😊	😊	😊	😊	😊
Image representation	😊	😊	😊	😊	😊	😊
Sparse explanation	😊	😊	😊	😊	😊	😊

Table 1: Indication of the weaknesses and strengths of the methods presented on a scale of 1 to 5.

Throughout this state of art, we firstly defined what is interpretability and cited some challenges and basics interpretable methods. Interpretability is a novel domain that is developing and counts on many recent works. Thereafter, we presented two families of methods: *linear explanations* and *rule based explanations*. Then, we went further by highlighting some issues in methods such as Lime and Anchors and some works that improve them. We noted that in some cases, state-of-the-art methods deliver poor explanations. For instance, explanations based on rules may be too complex since the length of the rule may be too long thus defining too specific instance. Some hypotheses claim that the quality of the explanations methods depends of the type of black box model.

The first step of this internship was to identify some of these limitations. This internship is a part of the ANR project FAbLe⁴ dedicated to automatic interpretability and we focused on solving some of the drawbacks of Anchors. In the followings sections we introduce further in details about Anchors and its internal mechanisms upon tabular and textual data. Thereafter, in section 5 we present our contributions to improve Anchors and in section 6 we discuss our experimental results. In the final section, we provide an outlook of the future work.

4 How to assess the quality of an anchors?

In this section we present further Anchors’ internal mechanisms and limitations. Thereafter we introduce the evaluation measures we use to assess Anchors and point out its limitations on tabular and text data.

4.1 Anchors further in details

We recall that Anchors [19] is a local agnostic method based on decision rule introduced by Marco Tulio Ribeiro *et al.*. As illustrated in figure 10b, there may have multiple possible linear explanations for a given use case. Thus, linear explanations are very unstable and are not the best way for user to generalize. As stated by Finale in [7] rule-based explanations are more comprehensible for most of human than linear explanations. As we can observe on figure 10b, there are multiple possible linear explanations and there is no way to know which one is the best adapted to mimic the black-box model. For instance, this is the case for the bottom line of figure 10b. The black box model in background will not predict the class *star* for the points in black that are above the bottom line. As mentioned in figure 7 for the LSTM explanation example, a user who has resorted to the bottom line explanation may apply this explanation for a wrong instance afterwards, because and contrary to Anchors, linear explanations do not provide a notion of scope for explanations. In this section, we present further details about Anchors and its usability on tabular and textual data. We also introduce the evaluation measures that are used for future experiments.

Anchors generates explanations based on decision rules that are induced from sampling instances. Anchors starts by a sampling phase that creates instances around the target instance in the interpretable feature space. It then uses the black-box model to classify these instances. Once the instances in the sample has been classified, Anchors learns decision rules to predict the outcome of the black box based on this sample. For the remainder of this report, the words “rule” and “anchor” are used interchangeably.

There is a crucial difference between the tabular and text data to generate perturbed neighbors. Applied to tabular data, Anchors samples instances from the training set. It then discretizes the

⁴Framework for Automatic Interpretability in Machine Learning

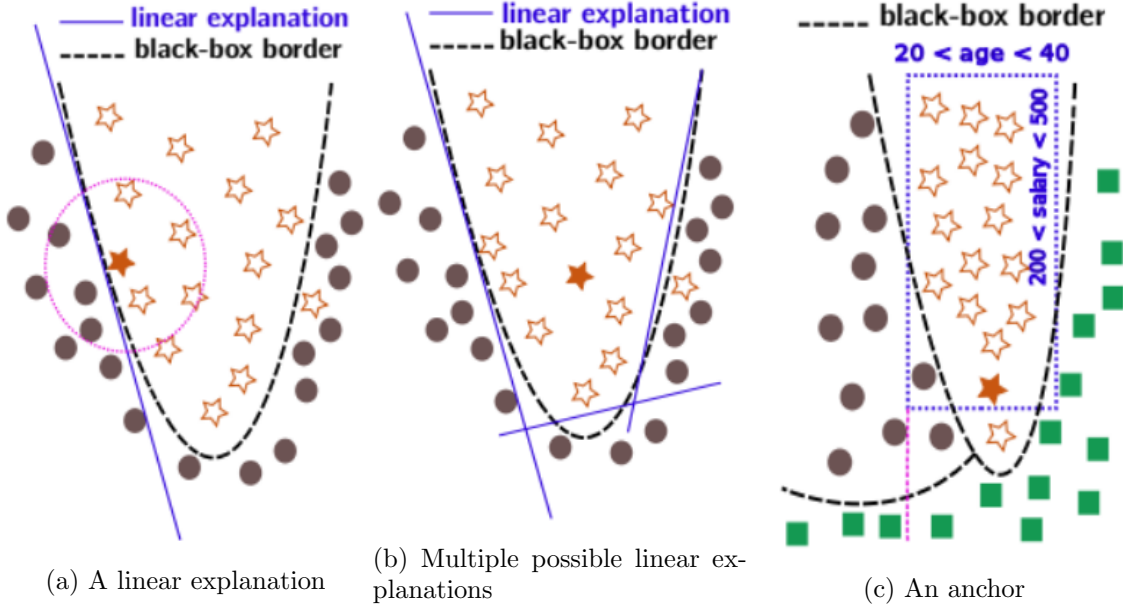


Figure 10: Comparison between linear and rule based explanations.

instances in the sample by associating the attribute values of the sample to a set of intervals learned with a discretization method. The intervals with a sufficient number of “good examples” covering the max are selected. A good example corresponds to an instance that have a feature value that validates the rule of the anchor. As a first definition, we can consider coverage as the ratio of good examples among all instances sampled. In section 4.2.1 we provide a more detailed definition.

We illustrate Anchors for tabular data by means of the Titanic dataset from Kaggle⁵ that predicts if a passenger survived or perished in the Titanic. The contributions on textual data are illustrated with the polarity dataset that is composed of textual comments for movies. The black-box model predicts if the comment conveys a positive or negative sentiment.

4.2 Evaluation Measures Applied to Tabular Data

Anchors applied to tabular data randomly selects instances from the training set with discounted draws. This means as we can observe on table 2 that Anchors may draw the same instances multiple times (*i.e.*: *instance number 12.*) This implies that even from a small training dataset, Anchors may generate randomly as many sample instances as it requires. After discretization of the numerical variables, it represents the instances in a matrix with each value associated to a feature as in the example table 2.

From all the sampled instances represented in this array, Anchors then generates a matrix of coalition vectors. These coalition vectors indicate for each feature if a sample instance has the same feature value as the target instance. For example, instance number **40** has the same feature value for the attribute “**Nb of parents**”. This induces in the matrix of coalition vector (see table 3) that they both have a **1 value**.

Now, we introduce some metrics that make use of the matrix of coalition vectors.

⁵<https://www.kaggle.com/c/titanic/data>

No instances	Age	Sex	Nb parents	Nb children	Class	Port	Survived
<i>target instance</i>	12	1	2	0	2	1	1
12	18	0	0	0	3	2	1
21	35	1	0	2	2	1	0
12	64	1	0	0	1	3	0
40	12	0	2	0	2	1	1
54	22	1	1	1	1	2	0
42	27	0	0	1	1	1	1

Table 2: Representation of different instances randomly picked from the training dataset.

No instances	Age	Sex	Nb parents	Nb children	Class	Port	Survived
<i>target instance</i>	1	1	1	1	1	1	1
12	0	0	0	1	0	0	1
21	0	1	0	0	1	1	0
12	0	1	0	1	0	0	0
40	1	0	1	1	1	1	1
54	0	1	0	0	0	0	0
42	0	0	0	0	0	1	1

Table 3: Example of matrix of coalition vectors for perturbed instances in tabular data. 1 correspond to data that have the same feature value as the target instance and 0 correspond to a different feature value.

4.2.1 Coverage

The notion of coverage on tabular data corresponds to the number of instances generated during the sampling phase that share the same feature value as the anchor compared to the number total of instances. Intuitively, it can be perceived as the proportion of generated instances that are covered by the rule r . For instance, given the anchor $r = \{(Nb\ Children = 0) \rightarrow survives.\}$, we can observe on table 3 that 3 out of the 6 generated instances have the same feature values as the target instance for the feature “Nb Children”, since they have a 1 in the matrix of coalition vectors. Thus, the coverage equals to 0.5 or 3/6 with 6 the total number of instances. Formally, coverage corresponds to this equation:

$$cov(r) = \frac{1}{n} \sum_{i=1}^n (x_i | \forall j \in R^f, x_{ij} = r_j) \quad (13)$$

In above formula, r corresponds to the rule generated by Anchors, n corresponds to the number of perturbed instances and x corresponds to an instance from the sampled set. R^f symbolizes the set of features f and j corresponds to the j^{th} feature in R^f .

4.2.2 Precision

The notion of precision on tabular data corresponds to the number of instances generated during the sampling phase that are covered by the anchor rule and share the same prediction. Intuitively,

precision represents the ratio of instances that validate the prediction of the rule among instances covered by the anchor rule. Consider again our example from table 2 which predicts if a passenger of the Titanic survived or not. Given the anchor $r = \{(Nb\ Children = 0) \rightarrow survives\}$. Three of the six generated instances have the same feature value as the target instance. Moreover, two out of these three instances have the same prediction “*survived*” as the target instance, meaning that the precision equals to $2/3$. Formally, we define precision as follow:

$$prec(r) = \frac{\sum_{i=1}^n (x_i | \forall j \in R^f, x_{ij} = r_j, y_i = y_{target})}{\sum_{i=1}^n (x_i | \forall j \in R^f, x_{ij} = r_j)} \quad (14)$$

In order to illustrate what is an anchor, we generated a virtual dataset in two dimensions with two different classes. This generated sample is implemented with the python function *generate blobs* from scikit learn⁶. There are 10k instances randomly generated. In the figure below, we can observe an anchor example based on this generated dataset. The anchor is represented by the red line and dashed lines corresponds to the direction of the anchor. The anchor rule is indicating on the top left and the purple star represents the target instance. The black box model is a decision tree learnt using scikit learn. On the background, black and white represents the black box prediction. Black corresponds to class 0 (or red class) predicted by the decision tree while white represents class 1 (or blue class) predicted by the complex model. From this toy example, we can deduce that the anchor rule mimics well the black box model and obtains a maximal coverage of 0.5. This happens because the classes are perfectly balanced. We can easily observe that the precision is also maximal and has a value of 1 since each point covered by the rule matches class 1 as the target instance represented by the purple star.

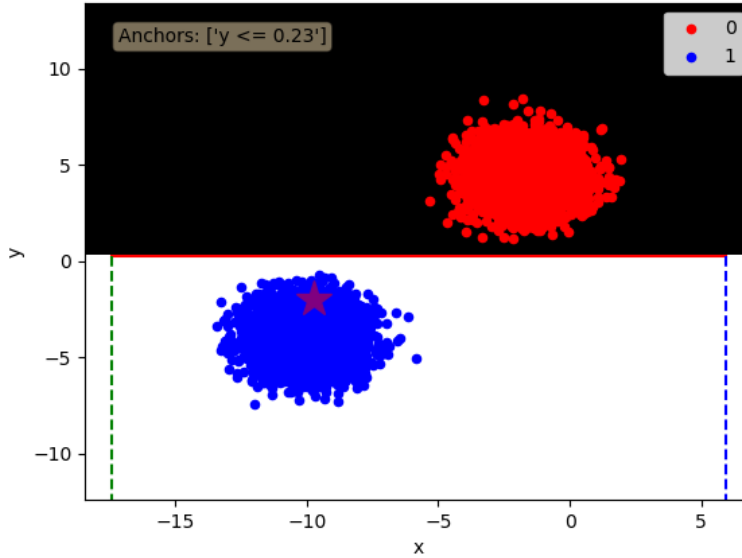


Figure 11: Example of an anchor on randomly generated data.

⁶https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

4.2.3 F1 Measure

As we lead experiments on multiple datasets with multiple black box models and multiple discretization methods, we also introduced a new metric inspired from the F1 measure. F1 measure (or F1 score) is the harmonic mean of the precision and recall. As mentioned in section 2.7, it is one of the most common metrics used by the machine learning community. We adapted the F1 score to our situation because (a) coverage is analogous to recall, and (b) the F1 provides a trade-off between these two metrics. Hence, we use it measure in order to facilitate comparison between discretization methods and black box models. Nevertheless, we had to adapt the coverage measure to compute the F1 score since coverage is bounded by class dispersion. Indeed, without normalization, coverage may never be equals to 1. Thus, in order to compute the F1 measure we first normalize the coverage value dividing each coverage value by the max coverage.

$$F_1(r) = \frac{2}{1/cov_{norm}(r) + 1/prec(r)} \quad (15)$$

4.3 Evaluation Measures Applied to Text Data

For explanations based on textual information, Anchors does not access the training data. Thus, it employs a natural language model that is train to replace words and find other similar words. By default, these models come from Spacy⁷. Thus, we suspect that the effect of Anchors on the black box depends of the chosen spacy model and on the black box training data. Indeed, some spacy models are trained on written web text (*blogs, news, comments*) while others are trained on Wikipedia for example. Depending of the black box model application, some spacy model may be more suitable than others.

Similarly to the tabular data, Anchors when applied to text data generates perturbed sentences. These sentences are represented by a matrix of coalition vectors where a feature value of 1 means that the word of the target sentence is kept while a 0 means that it is replaced. When using Anchors, there are 2 possibilities: replace words with a mask word, *i.e.*: a word with no impact on the sentence. Or replace with a word of similar meaning *i.e.*: a “book” is replaced by “article” or “movie”.

No instances	determinant	verb	determinant	adjective	noun	punctuation	sentiment
target instance	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
2	0	1	0	0	0	1	1
3	0	1	1	1	1	1	1
4	1	1	1	0	0	1	0
5	1	1	1	1	0	1	1
6	0	1	0	0	0	1	0

Table 4: Example of matrix of coalition vectors for perturbed instances on textual data.

⁷<https://spacy.io/>

Before replacing words from the target sentence by other words with similar meaning or masked words, anchors generates randomly a matrix of coalitions vectors. In this matrix each word from the target sentence represents a feature and a value of 1 means that the word is kept while a value of 0 means that the word has been replaced. Words are categorized according to their part of speech tagging. Some grammatical classes are replaced such as nouns, verbs, adjectives, adverbs, determinant and ad position while other tagging like punctuation or pronoun are never replace. This involves that Anchors acts only on these tags to replace words thus is not able to return other words for the rule. Last but not least, Anchors gives a special treatment to the verb “be” and preposition tags and never changes them.

No instances	determinant	verb	determinant	adjective	noun	punctuation	sentiment
target instance	This	is	a	good	book	.	pos
1	both	is	an	good	book	.	pos
2	there	is	some	beautiful	movie	.	pos
3	both	is	an	good	book	.	pos
4	this	is	a	bad	photo	.	neg
5	this	is	a	good	text	.	pos
6	every	is	any	wicked	textbook	.	neg

Table 5: Example of perturbed instances on textual data with replace word.

In this section we introduce the version we use to compute coverage and precision for our experiment. The computation method we used is based on the training dataset while in the original computation method, precision and coverage are compute on samples sentences. We decided to change this method since we consider that precision and coverage represents better the real situation and are better metrics use on the training data rather than samples instances when it is possible of course.

4.3.1 Coverage

Originally, coverage is computed based on the matrix of coalition vector generated by Anchors. For each perturbed sentence, it corresponds to the proportion of instances that contains the anchor rule (*each words present in the rule.*) The idea is that coverage designates the ratio of sentences that contains each words of the anchor among all sampled sentences. In the table 5, if the anchor corresponds to the presence of words “good” and “book” in the sentence, coverage equals to 2/6. This is due to the fact that these words are present simultaneity only in sentences 1 and 3. The new version to compute coverage that we introduce is similar but is based on the test dataset instead of the perturbed instances generated by the natural language model used by Anchors.

$$cov(r) = \frac{1}{n} \sum_{i=1}^n (x_i | \forall j \in R^f, x_{ij} = r_j) \quad (16)$$

Where r corresponds to the anchor rule, n corresponds to the number of sentences composed in the training set or the sampled generated (*matrix of coalition vectors.*) x_i corresponds to the sentence number i and x_{ij} corresponds to word at position j in sentence i . R^f represents the interpretable feature space in this case, the set of all words.

4.3.2 Precision

As for tabular data, textual explanations computes precision based on the label of the target instance. Intuitively, precision corresponds to the proportion of instances among instances covered by the rule that share the classification of the target sentence. For instance, in the table 5 the label is “*positive*” for the target sentence. For each perturbed sentence that contains each words of the rule, it computes the proportion of sentences that have the same label as the target sentence. For the table 5, we can observe that precision equals to 1 as all elements that contains words “*good*” and “*book*” are predicted “*positive*” by the black box model. Formally, the precision is compute as in formula 17 where y corresponds to the label of the instance.

$$prec(r) = \frac{\sum_{i=1}^n (x_i | \forall j \in R^f, x_{ij} = r_j, y_i = y_{target})}{\sum_{i=1}^n (x_i | \forall j \in R^f, x_{ij} = r_j)} \quad (17)$$

4.3.3 F1 Measure

The F1 measure is implemented in the same way as for tabular data (see Section 4.2.3). In order to be consistent with our approach on tabular data, we adapted the F1 score to our situation on textual data. For similar reasons, there are multiple datasets and multiple black-box models to compare, we introduced this F1 measure. We recall that we are able to adapt F1 score to our situation since coverage is analogous to recall. Moreover, F1 provides a compromise between precision and recall and we do not want to choose between precision and coverage. As mention for tabular data, since coverage is bounded by class dispersion, we had to adapt the coverage measure to compute the F1 score. Hence, in order to compute the F1 measure we first normalize the coverage value.

$$F_1(r) = \frac{2}{1/cov_{norm}(r) + 1/prec(r)} \quad (18)$$

5 Contributions

In this section we present our contributions that highlight some of limitations of Anchors and propose improvements. First, we focus on tabular data and present the impact of the discretization method on the quality of Anchors. Then we introduce our new method to improve Anchors on text data using pertinent negatives.

5.1 Impact of Discretization on Anchors Tabular

Anchors on tabular data must have access to training data in order to generate meaningful conditions, since predicting the distribution for features such as “age” or “sex” without any information remains a very challenging objective. Discretization is a process that can only be apply to tabular data. Hence, we focus on the impact of discretization on Anchors for tabular data only.

Features in a learning problem may be categorical or numerical. Numerical refers to features taking on integer or real values. In this section we focus on numerical attributes that must be discretized to appear as conditions in anchors, such as *age*, *income*, *size*, *etc...* Discretization is the process of mapping the values of a continuous variable to a set of categorical values. There exists different methods for discretization.

In Anchors by default there are 3 methods: Quartile, Decile and Entropy. Quartile divides the domain of a feature into 4 sub partitions or intervals using the first, second, and third Quartiles of the distribution of values as cutting points. Decile is a similar method that divides the partition into 10 sub partitions instead of 4. Entropy is a method based on the entropy principle that measures the unpredictability of the information content in a message source. It uses the principle of maximum a posteriori since it employs training data and corresponding labels generated by the black box to find the best number of intervals and their corresponding values. Entropy method generates intervals that have a low variance of classes predicted by the black box. The idea is that entropy methods generates “pure intervals” whose all instances present in each interval belong to the same class.

We study on an artificial dataset the impact of discretization by displaying the difference between an anchor based on the **entropy** method and an anchor based on the **Quartile** method. As we can observe on Figure 12 an anchor based on the Entropy method mimics better the black box model represented in the background than the one based on Quartile method. This toy example illustrates the impact of the discretization method on the quality of anchors. We evaluate this impact in detail in Section 6.4.1.

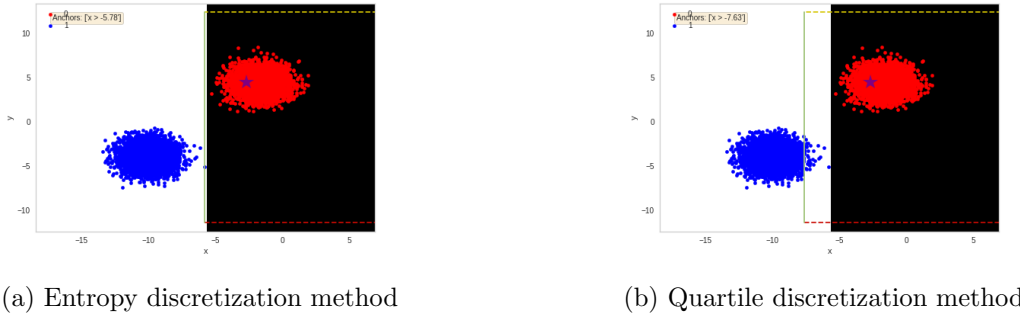


Figure 12: Impact of the discretization method on the quality of the anchor. Black and white in the background corresponds to the black box model prediction for class red or blue. On this generated example there are 2 features x and y and the anchor is represented by the green line while dashed lines correspond to the direction of the anchor application.

In order to analyse the impact of the discretization method, we integrate two novel discretization methods. The first one is k-means and represents a baseline due to its simple form; the second is MDLP [8]⁸. We explain both methods in the following sections.

5.1.1 K-means

K-means [11] is a method that aims at partitioning n observations into k clusters. Each observation belongs to the cluster with the nearest cluster centroid. It is a very popular method for cluster analysis in data mining since it is very easy to implement from the number k of clusters given by the user. Given a set of observations (x_1, x_2, \dots, x_n) , where each observation is a d -dimensional real vector. K-means clustering aims at partitioning the n observations into k (with $k \ll n$) sets $S = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within-cluster variances. Formally, the objective is to

⁸Minimum Description Length Principle

minimize the following equation where μ_i is the mean of points in S_i :

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \mu_i\|^2 = \arg \min_{\mathbf{S}} \sum_{i=1}^k |S_i| \text{Var } S_i \quad (19)$$

This algorithm is composed of 4 steps:

- Step 1: Initialize k clusters centers that are randomly generated within the data domain.
- Step 2: k clusters are created by associating every observations with the nearest mean.
- Step 3: The centroid of each of the k clusters becomes the new mean.
- Step 4: Steps 2 and 3 are repeated until the convergence criteria is reach.

One of the main drawback of k-means is the choice of k . Hence, it is difficult to select the best number of clusters in which data is divided. Thus, we employ the elbow method [21]⁹ to pick the best k . The “best” k corresponds to the one that has a lower distortion score for a smallest fit time and highest number of centroid. Distortion score is defined as the sum of the squared distances between each observation vector and its dominating centroid.

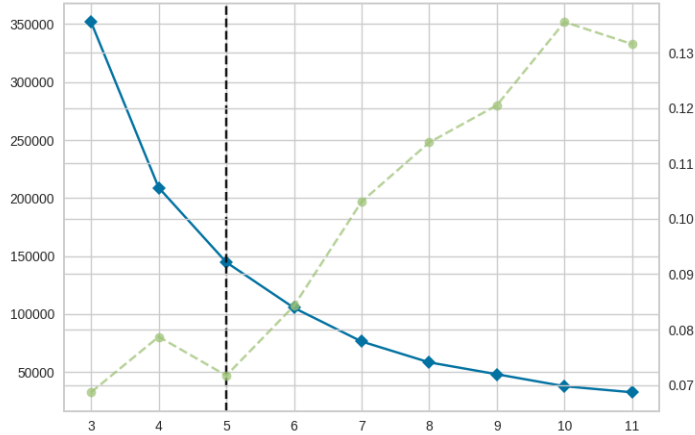


Figure 13: Identification of Elbow point on the “Age” feature. x axis corresponds to the value of k . y axis on the left depicts the distortion score and the y axis on the right represents the time to fit in seconds. The black dash lined corresponds to the best value of k returned by the Elbow method.

Elbow method is the oldest method for determining the best number of clusters in a data set. Intuitively, the idea is to start with a small value of k (*i.e.*: $k = 3$) and keep increasing the value of k by 1 in each step. For each step, this algorithm computes k-means with the k associated (*i.e.*: $k = 3$) along the cost that comes with the training. At some value for k (*i.e.*: $k = 5$) the cost drops significantly, and thereafter it reaches a plateau when it still increases the value of k . The desired k value is reached just before the plateau as in figure 13.

⁹<https://www.scikit-yb.org/en/latest/api/cluster/elbow.html>

5.1.2 MDLP

MDLP is a method based on entropy that computes the maximum a posteriori construct on the Bayesian principle. Intuitively, Minimum Description Length Principle returns the minimal number of “pure” intervals needed to separate instances from distinct classes. Compare to a traditional entropy method, MDLP focuses on compression minimality, hence it will always output as few intervals as possible. Moreover, it ensures that it always returns a value like $B1$ or $B2$ on figure 14 for the interval value. These values correspond to a border value and are preferred to a compromise value. A compromise value corresponds to a potential cut point between two border values that is wrongly chosen by the entropy method. Usual entropy method may choose wrong value as in figure 14 since it minimizes the weighted average entropy. A border value is define in definition 4 in [8] as:

“A value T in the range of A is a boundary point iff in the sequence of examples sorted by the value of A , there exist two examples $e_1, e_2 \in S$, having different classes. such that $A(e_1) < T < A(e_2)$; and there exists no other example $e' \in S$ such that $A(e_1) < A(e') < A(e_2)$. If T minimizes the measure $E(A, T; S)$, then T is a boundary point.”

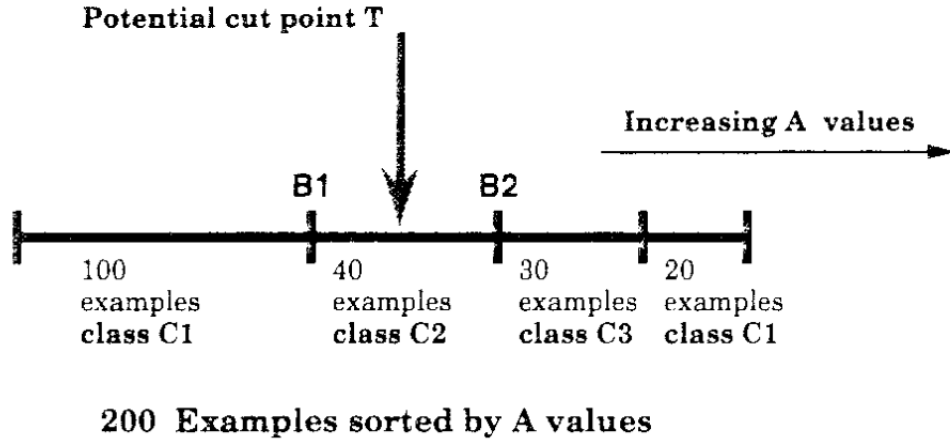


Figure 14: Image from [8] that compares how a traditional entropy method works by cutting within a class in order to minimize the weighted average entropy of the two sets instead of cutting on a border value like $B1$ or $B2$.

As MDLP aims at returning only boundary points, thus it allows to run faster and with assurance that the intuition of a good discretization method is ensured. Experimental results are shown in section 6.4.1.

5.2 Improve Anchors Text Quality with Pertinent Negatives

By using Anchors, we discovered that often, there is no rule that are above the threshold of 95% of precision fixed by Marco Tulio Ribeiro in [19]. In such cases, Anchors returns a rule that has a lower precision. We argued that this kind of rules may not be sufficient depending of the context. For instance, a computer scientist using a neural network to translate a speech of Donald Trump from English to Korean. He uses Anchors to ensure White House that his neural network will not send wrong message to Kim Jong Un. Anchors is not able to find sufficient rules with a precision

nice	film
nice	movie
bad	film
bad	movie
hilarious	comedy
science	film

Table 6: Bi-gram matrix example based on movies comments after tokenization, lemmatization and removal of stop words.

threshold of 95% so he must uses a different version of Anchors.

In this part we introduce our improvement for Anchors on text data. We propose a novel method of explanation based on the work from [6] that employs pertinent negatives to explain a black-box model. Pertinent negatives correspond to elements whose absence impact the most the prediction. It is an element (*i.e: a word in the case of textual data*) such that his presence would change the prediction of the black box.

We implement what we call the *pertinent negatives* method by creating a bi-gram matrix of words from the training dataset. A bi-gram matrix is computed based on the frequent association of two words following one each other. For instance, in a dataset containing movies comments, we frequently observe the word “movie” next to “nice” or “bad” (as depicted in figure 6.)

We clean this dataset with classical natural language processing steps such as:

- **Tokenisation**, which means to split a document into smaller units called tokens such as individual words.
- **Lemmatization**, that corresponds to the process of grouping together the inflected forms of a word so they can be analysed as a single item. Lemmatization is often confused with stemming since both methods have the same purpose: reducing the form of each word into a common base or root.
- **Removal of “stop words”**. Stop words refers to the most common words in the language. Hence, we remove them since they are so frequent that they are consider as useless. For instance: “the”, “a”, “this”... are not impacting classification.

We keep adverbs when removing stop words since by definition (*an adverb alters or modifies the meaning of the verb or noun attach to it*) adverbs are the most important for pertinent negatives. Hence, the purpose of pertinent negatives is to detect frequent association of words that impact the most the prediction. Thus, adverbs like “not” or “slowly” may impact a lot. For the experimental results we only build a co-occurrence matrix based on sentences from the polarity dataset presented in section 6.1 whereas if the training set is available, pertinent negatives can adapt its co-occurrence matrix and pre-process any other training dataset.

Afterwards, pertinent negatives generates as in traditional Anchors, different sentences with most frequent bi gram words that are present in the target sentence “*This is a good article*” as in the table 7. In this table, “not” is a frequent bi gram word associated with “is” while “scientific” is related to “article.” As we can observe on this example, “not” is the adverb from the set of pertinent negatives words that impact the most prediction. Indeed, every time “not” is present;

black-box model changes the prediction from his initial prediction: **positive** to a different one: **negative**.

This	is		a	good		article	positive
This	is	not	a	good		article	negative
This	is		a	good	scientific	article	positive
This	is	not	a	good	scientific	article	negative

Table 7: Table with new sentences generated to find pertinents negatives with most frequent bi-gram words. Words generated from the bi-gram matrix are represented in bold and are not present in each line. Prediction of the model is the final column and is either "*positive*" or "*negative*".

For the moment pertinent negatives is a different method than Anchors. One of our purpose is to extend Anchors to employ pertinent negatives when Anchors is not able to produce rules with a sufficient threshold.

During this internship, we aim at solving the question: "When are anchors based explanations not a good idea?". Guided by this question we discovered two ways to improve Anchors. Firstly, we studied the impact of the discretization methods on tabular data. Hence, we figured out that anchors based explanations are not a good idea when the discretization of the numerical variables does not consider the distribution of the black box answers. Thus, we implemented two novel discretization methods: k-means and MDLP. Secondly, we implemented a novel mechanism of explanations in the name of pertinents negatives. It resolves cases in which anchors based explanations are too specific resulting in a poor confidence threshold. Pertinents negatives is an extension of the traditional way Anchors works on textual data by adding the absence of frequent bi gram words in the sentence. It permits by the non presence of these words to explain the prediction of the complex model.

6 Experimentations

We evaluate our contributions on different datasets and contrasting black-box models. We separate results in two parts: tabular and textual results. But first, we start presenting all the datasets that are used for experiments. Thereafter, we introduce each complex models that are used to be explained, to conclude, we comment our experiments results.

6.1 Datasets

In this part, we will first introduce all different datasets that are used to implement and test Anchors. The first one we already presented in section 4.2 is **Titanic** downloaded from Kaggle¹⁰. The second one is named **Adult** and aims at predicting if a person earns more or less than 50k\$ based on several characteristics such as age, marital status, gender, or occupation. The features are a mixture of ordinal and categorical data. This dataset is one of those used to evaluate Anchors [19]. We also create three artificial datasets that are represented in two dimensions with two possible classes. One generates two distinct blobs as depicted in figure 15a, and the other two distinct moons as shown in figure 15c and the last two distinct circles illustrated in figure 15b. From now on, we will respectively call them **generated blobs**, **generated circles** and **generated moons**.

¹⁰<https://www.kaggle.com/c/titanic/data>

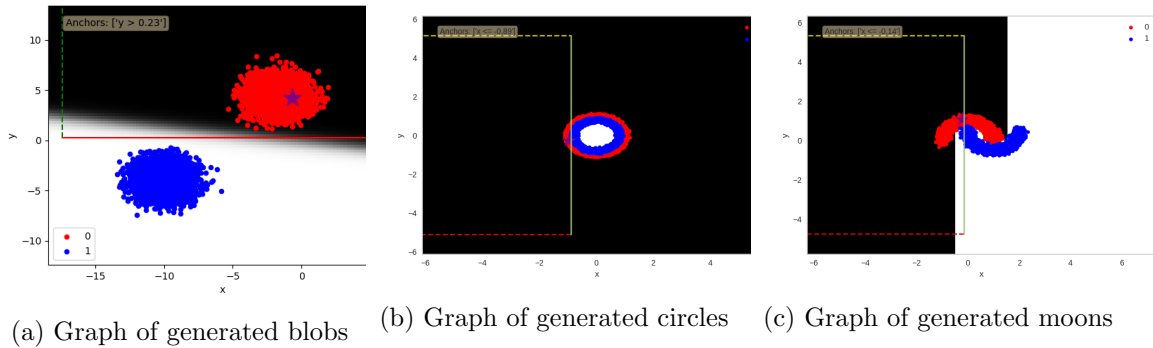


Figure 15: Graphs representing three generated artificial datasets.

We introduce these datasets for different reasons. We experiment on **Titanic** and **Adult** in order to produce explanations upon complex and more realistic black-box models. Subsequently, we decided to generate artificial datasets to visualize data and show intuitively how an anchor rule looks like. Hence, this allows us to modify data and the complex model in background at no cost. As depicted on figure 12, graphical results confirm our intuition that discretization methods do impact the quality of the rule.

For textual data we train black-box models upon the **polarity** dataset that consists of 10000 commentaries of users on movies. The purpose of the black-box model is to predict the sentiment (*i.e.* *positive* or *negative*) of the comment. It consists of 5000 positive and 5000 negative comments and is available on Cornell web page¹¹. A similar dataset is the one used to train the Bert black-box model [5], that predicts the sentiment of a sentence¹². Finally, last dataset is **Tweet** [2]. Since the conditions of use of Twitter no longer allow the dissemination of dataset on the Internet, we encourage to create a Twitter developer account and get all the data and scrawler need on this link¹³ in order to reproduce these experiments. These datasets has two main advantages. First, they are very large dataset (more than 400k tweets.) Secondly, tweets are available in Spanish and English thus it permits to employ Anchors on different spacy models train on different languages. We based our experiments on work presented in [1]. In this work, the black-box model predicts which of the 20 most popular emojis (as illustrated in figure 16) was originally in the tweet. Thus, this dataset is the only one we used to predict multi class.

To resume, we experiment over 5 tabular datasets: **Titanic**, **Adult** and three generated datasets: **generate blobs**, **generate moons**, **generate circles** and two text datasets: **Polarity** and **Tweets**.

6.2 Black-box Models

Driven by the hypothesis that Anchors quality depends of the type of the model in background, we compute and test Anchors on multiple black-box models. Most of these black boxes are *scikit learn*¹⁴ implementations such as random forest, support vectors machines, multi-layer perceptron,

¹¹<http://www.cs.cornell.edu/people/pabo/movie-review-data/>

¹²<https://github.com/clairett/pytorch-sentiment-classification/raw/master/data/SST2/train.tsv>

¹³https://competitions.codalab.org/competitions/17344#learn_the_details-data

¹⁴<https://scikit-learn.org/stable/>





















									
100.7	89.9	59	33.8	28.6	27.9	22.5	21.5	21	20.8
									
19.5	18.6	18.5	17.5	17	16.1	15.9	15.2	14.2	10.9

Figure 16: Top 20 tweets and their corresponding frequencies.

decision tree, logistic regression and multinomial naive bayes.

6.3 Details Implementations

We run Anchors on these different datasets with multiple black-box models and a precision threshold of **0.95**. The experiments were run on a cluster from the Inria lab. All the code is available on github¹⁵ and each dataset is available online. All the graphs presented in this report are available on github along some others such as graphs with different precision threshold and other complex models.

6.4 Results

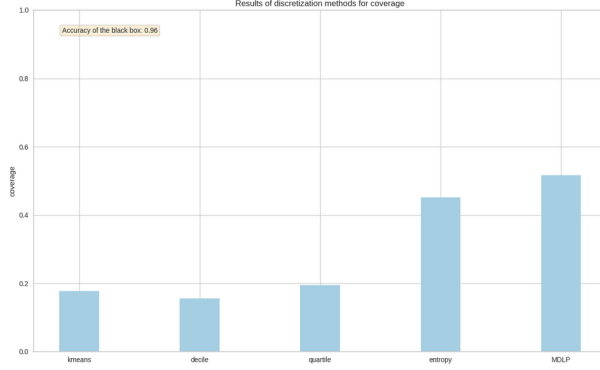
We introduced in section 4.2 and section 4.3 respectively the evaluations measures for tabular and textual data that we need to understand the following graphs. In the first part we will present tabular results and then concentrate on textual results.

Figure 17 and 18 illustrate the impact of the discretization method on Anchors, upon a logistic regression on the generate circles dataset with a precision threshold of 95%. We present similar graphs for each black-box models at the three different threshold for each datasets. We also show comparable graphs representing the impact of the method to replace words in the target sentence for text data. In this section, we focus on Anchors with a precision threshold of 95% since this threshold is the baseline introduced in [19] for a sufficient anchor rule. Hence, we argued that the objective of an anchor is to explain a complex model with high precision. For this reason, we present only numerical results for this threshold while graphs for others threshold are available on github.

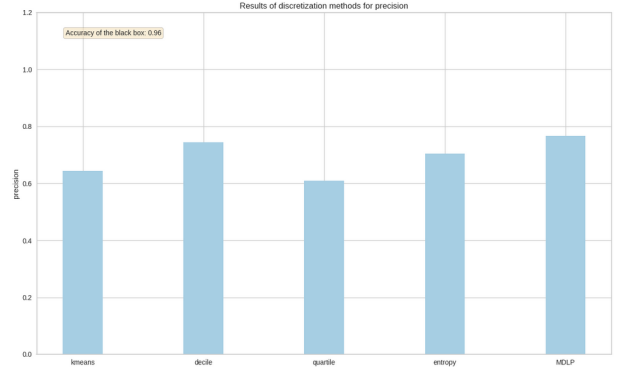
6.4.1 Impact of Discretization on Tabular Data

For each tabular dataset, we evaluated Anchors on: a **decision tree**, a **support vector machine**, a **logistic regression**, a **multi layer perceptron** and a **random forest**. To measure the impact of discretization methods, we employ five different metrics, three of them presented in section 4.2, namely, coverage, precision, and F1 as well as, runtime and the size of the explanations. Hence, we believe that a good discretization method reduces the size of the explanation, thus inducing rules easier to understand for a user. Moreover, the choice of the discretization method should not reduce the speed of Anchors but, as expected rather accelerate it.

¹⁵<https://github.com/juliendelaunay35000/anchors/tree/master/>

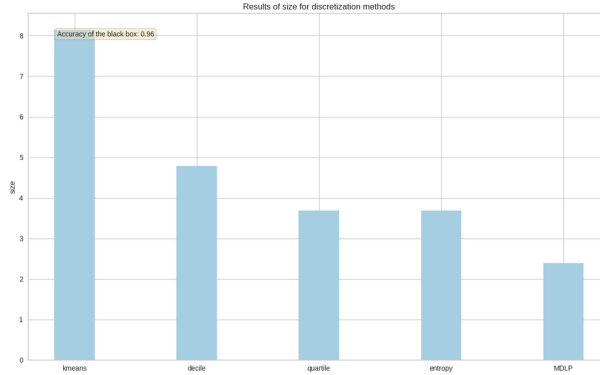


(a) Graph of coverage.

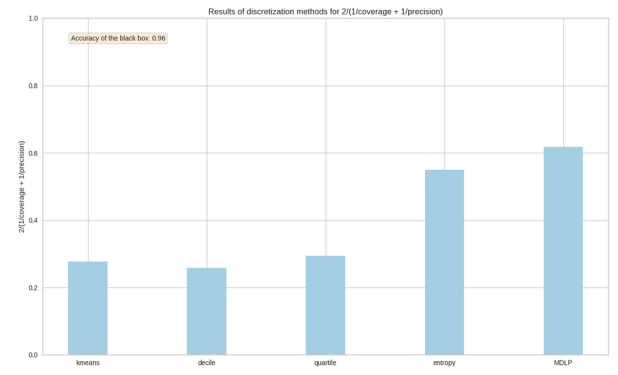


(b) Graph of precision.

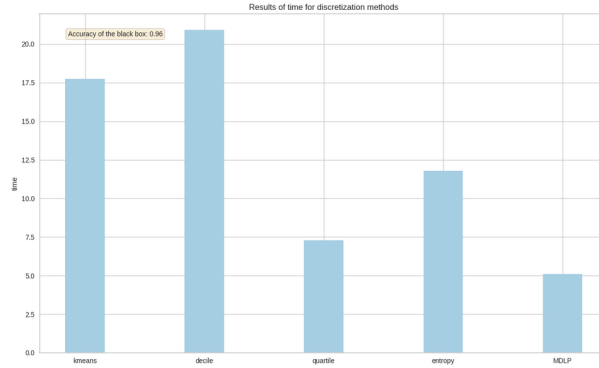
Figure 17: Graphs representing results for multiple discretization methods for Anchors. The order of discretization methods is: k-means, Decile, Quartile, Entropy and MDLP.



(a) Graph of size.



(b) Graph of F1 measure.



(c) Graph of time.

Figure 18: Graphs representing results for multiple discretization methods for Anchors. The order of discretization methods is: k-means, Decile, Quartile, Entropy and MDLP.

We compared five black-box models on five generated datasets with a precision threshold of **0.95** for Anchors. On Figure 19 we illustrate the comparison based on the F1 measure introduced in section 4.2.3. We present for each dataset, the F1 score obtained for each discretization method

and black-box model. F1 score is a measure bounded between 0 and 1. We recall that it corresponds to the harmonic mean of the precision and normalized coverage. As we observe, MDLP is always one of the two methods that obtains the best F1 score (*higher F1 score is better*). Depending of the dataset more than the black box models, Quartile and Entropy appear to be better discretization methods than k-means and Decile. Indeed, Quartile is a better discretization method than Decile and k-means based on the F1 score. Hence, F1 score favors a good precision and simultaneously a good coverage. Since Quartile creates fewer intervals, the coverage for each of these intervals is bigger than for Decile for instance.

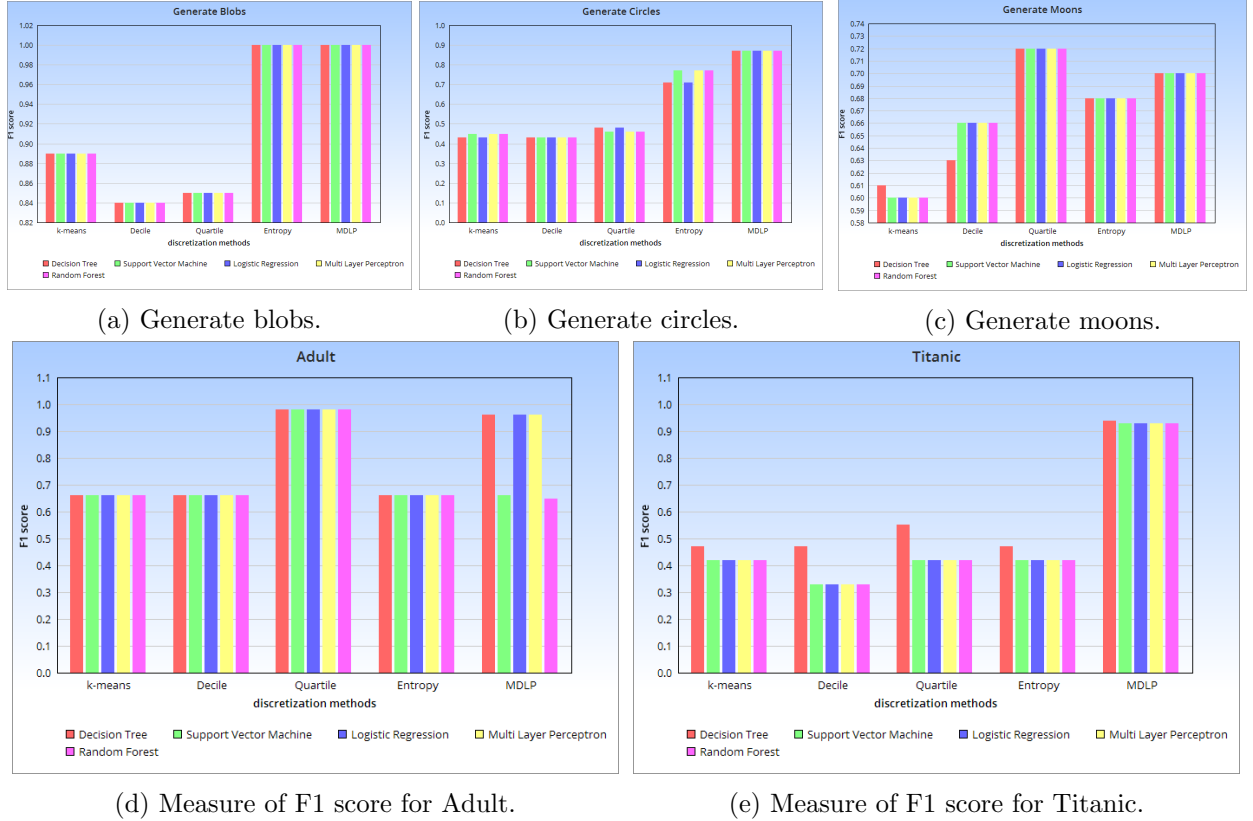


Figure 19: Graphs representing results for the normalized F1 score on Anchors for a precision threshold of 95% for the five different datasets and five different discretization methods. The order of discretization methods is: k-means, Decile, Quartile, Entropy and MDLP.

In Figure 20 we compare the impact on runtime of the different discretization methods. As we can observe, MDLP is 19 out of 25 times the method that makes Anchors run faster. In particular, MDLP is the fastest for all generated datasets while Decile is the slowest method. Entropy and Quartile methods vary more depending of the dataset. On the two more complex datasets (*i.e.* *Adult* and *Titanic*), the runtime is similar for MDLP, Quartile and Entropy. Hence, Quartile is the best on *Adult* dataset while MDLP is the best for the *Titanic* dataset.

We also compare the size of the resulting Anchors depending of the discretization method. We present only the size for the more complex datasets since for instance, on generate blobs, each method returns explanations of size one since the two classes are too easily separated. We can

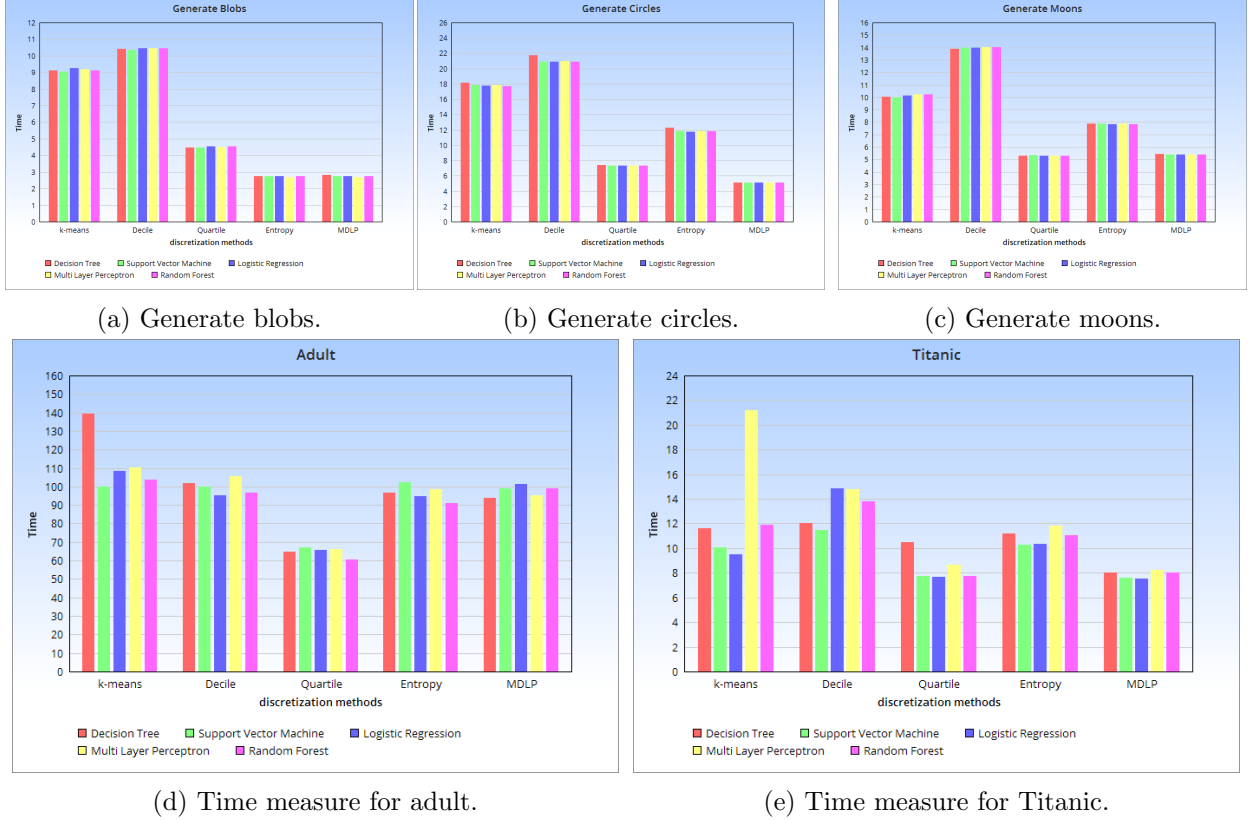


Figure 20: Graphs representing results for the time to generate explanations on Anchors for a precision threshold of 95% for the five different datasets and five different discretization methods. The order of discretization methods is: k-means, Decile, Quartile, Entropy and MDLP.

observe that on Adult, Quartile returns shortest rules, followed close by MDLP and then Decile and Entropy. For the generate circles dataset, MDLP returns shortest anchors followed by Quartile and Entropy. K-means returns longest rules. On the Titanic dataset, we noticed that MDLP outputs smallest rules for each black-box models, closely followed by Quartile, Entropy and then k-means. We observe that logistic regression is prone to long explanations. Explaining a logistic regression rather than a decision tree for instance increases a lot, moreover, it impacts more Decile and Entropy discretization methods than MDLP and Quartile. As future work, we envision to study the properties of black-box models that lead to long anchors.

As a final result, we discovered that regardless of the black-box model, if the complex model has an accuracy of 100% (*i.e.* *model predicts always the right class for any instance*), the quality of the anchor is the same for each model (*i.e.* *whatever discretization method is used, every anchor explanation model has the same coverage, precision, F1 measure and size*). We also observe that Anchors is stable on decision trees. This is not the case for the other black-box models, we argue that the simplicity of decision trees may be the principal reason.

Results presented in this section demonstrate that the discretization method has an impact on the quality of the Anchors. Among the studied discretization methods, MDLP has most of the time better results on the F1 measure and the length of the explanation while maintaining a maximal

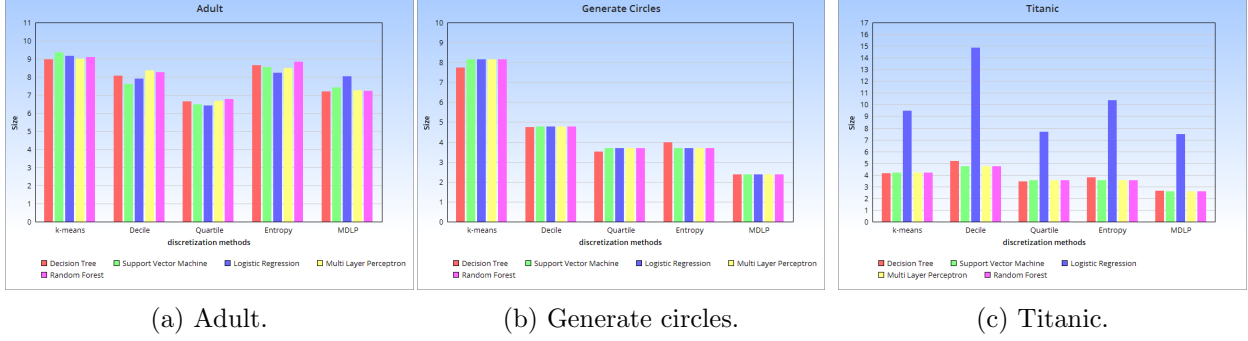


Figure 21: Graphs representing results for the size of explanations returned by Anchors for a precision threshold of 95% for three different datasets and five different discretization methods. The order of discretization methods is: k-means, Decile, Quartile, Entropy and MDLP.

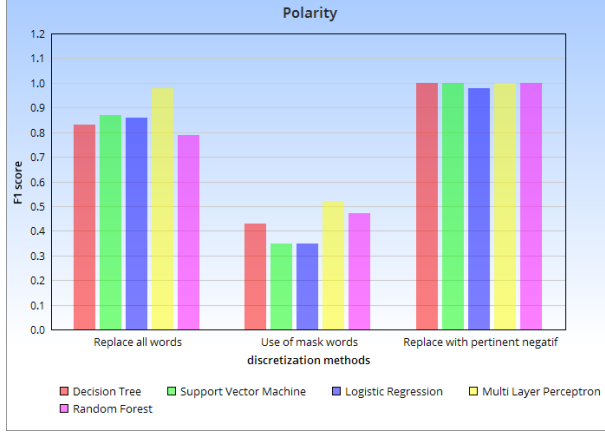
speed compare to other methods.

6.4.2 Textual Results

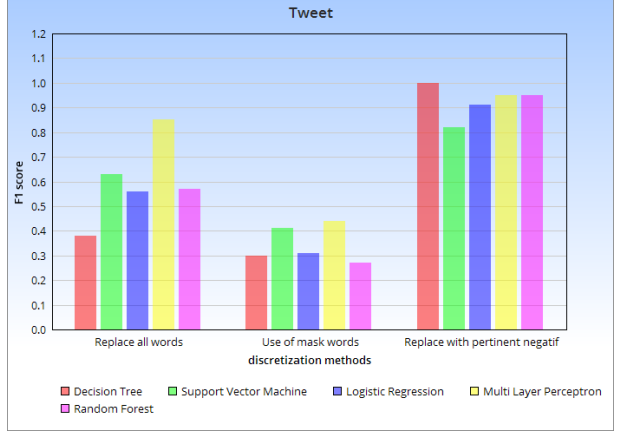
We evaluate our contributions for textual data on two datasets: **Polarity** and **Tweet**. We also evaluated it on the same five different black box models: **decision tree**, a **support vector machine**, a **logistic regression**, a **multi layer perceptron** and a **random forest**. We implemented Anchors on three different threshold: **0.2**, **0.75** and **0.95** whereas for the reasons mention before, we present only the results for Anchors with a threshold of 95%.

On figure 22, we can observe the results of the normalized F1 score for the comparison between the three methods to replace words. We recall that the first method replaces words in the target sentence with similar words generated by a spacy model. The second method uses mask words that have no impact on the black-box model for the prediction. The last method is the one we introduced and is named pertinent negatives. It adds words that do not appear in the target sentence but that are frequently associated in the dataset with other words from the target sentence. Thereafter it evaluates the impact of absence of these new words on the classification. Figure 22a suggests that use of mask word is clearly the worst method. Hence, the F1 score is around 0.4 that is two times less than the method that replaces words. The method that obtains the best F1 score (*approximately 1 for each complex model*) is pertinent negatives. On figure 22b, results are similar whereas the difference between pertinent negatives and replace words is accentuated. We can observe that Decision Tree results in the worst score for the first method while it increases the F1 score for pertinent negatives.

As for tabular data, we now analyze the results for the runtime of Anchors depending of the method to replace words. On figure 23a, we show that replacing with a mask words brings a speed-up of 10x on the polarity dataset when compared to replace words. The speed of Anchors based on pertinent negatives depends more on the black box. Hence, we can observe that based on a decision tree, pertinent negatives is two times faster than replace all words while for support vector machine and logistic regression they are similar. For the tweet dataset, as shown on figure 23b, the mask words method is again the fastest except when explaining a random forest. The method that replaces words is the slowest and is particularly slow with Anchors on random forest. The runtime of Anchors with Pertinent negatives depends on the black box. For instance, it is faster with a



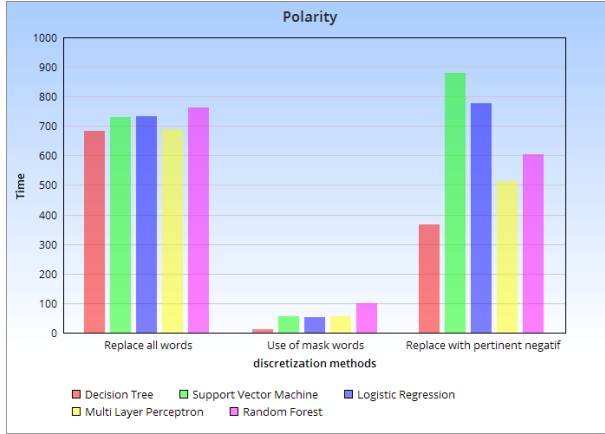
(a) F1 measure for Polarity.



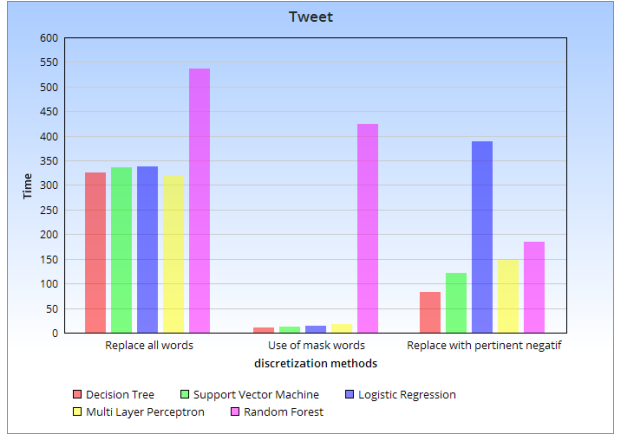
(b) F1 measure for Tweet.

Figure 22: Graphs representing results for F1 score on Anchors for a precision threshold of 95% for the two different datasets and three different methods to generate words. The order of methods is: replace words, use of mask words, replace with pertinent negatives.

decision tree and support vector machine whereas very slow with a logistic regression. Pertinent negatives is the fastest method to produce explanations on random forest for the tweet dataset.



(a) Time measure for Polarity.

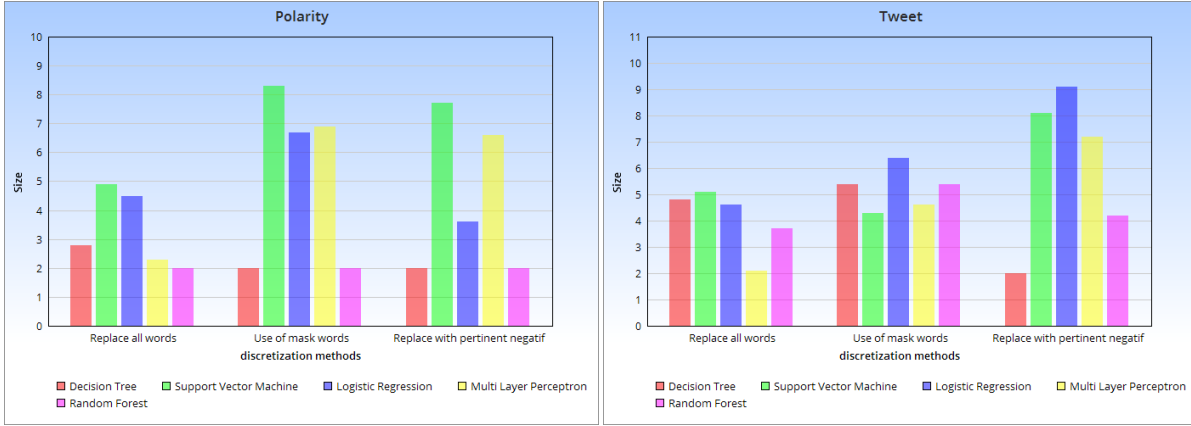


(b) Time measure for Tweet.

Figure 23: Graphs representing results for the time to generate explanations on Anchors for a precision threshold of 95% for the two different datasets and three different methods to generate words. The order of methods is: replace words, use of mask words, replace with pertinent negatives.

We also study the mean explanation size of the different replacement methods. On both datasets and for each method, we observe that the black-box models in background impacts the size of the explanations. Hence, on figure 24a, explanations for decision tree and random forest are the shortest for mask words and pertinent negatives. Indeed, on random forest each method returns explanations of size two (*including two words*). Pertinent negatives returns shortest explanations over logistic regression, and similar size for support vector machines and multi-layer perceptron than the method

that replaces with mask words. The method that replaces words by similar content is the shortest for support vector machines and multi-layer perceptron. On average, it returns explanations of less than five conditions.



(a) Mean size of explanations for Polarity.

(b) Mean size of explanations for Tweet.

Figure 24: Graphs representing results for the size of explanations on Anchors for a precision threshold of 95% for the two different datasets and three different methods to generate words. The order of methods is is: replace words, use of mask words, replace with pertinent negatives.

We introduced pertinent negatives, a novel explanation method for text data. Through experimental results, we argued that this method may extends Anchors thus defining an extending space to improve the precision and coverage. Hence, this novel method returns explanations with better F1 measure while preserving a good speed. Nevertheless, the length of the explanations are often longer than for other methods. Moreover, the choice of the complex model impacts more the quality of the explanation based on pertinent negatives than for the two other methods.

7 Future Work

In this section we sketch a novel idea to improve Anchors on tabular data using the basis of the works from [12] that we call *Growing Spheres*. Growing Spheres is a post-hoc interpretability method that seeks for the closest boundary counterfactual example. Intuitively, it aims at returning an artificial example that is closest to the target from a distinct class. Growing Spheres returns an artificial example that has a minimal number of change in term of features number and values by generating randomly instances around the target until it finds the border frontier of the black box. Growing Spheres makes no assumption about the data neither the black box model and is composed of two steps: the first one is the sample generation and the second one is the feature selection in order to returns simplest explanations. These two steps are represented on image 25.

During step 1, it generates artificial instances in the area of a sphere until it finds an instance from a different class. In order to do so, Growing Spheres generates instances uniformly distributed over sub-spaces representing spheres. Thereafter, there is 2 possibilities:

- In first case, there is no instance from a different class in the area of the sphere. Thus, the sphere is too small and need to be extend.

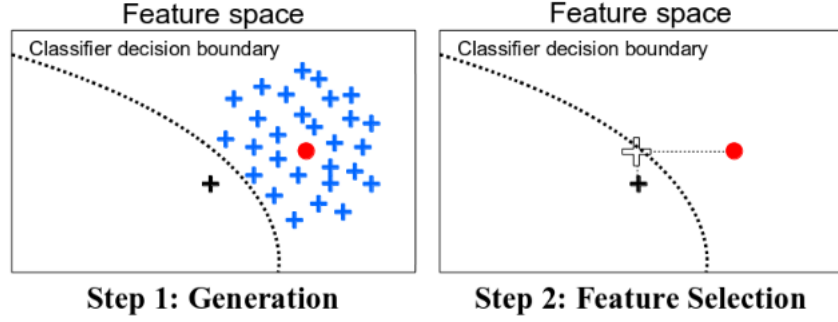


Figure 25: Image from [12] that illustrates the way Growing Sphere method is working. The red circle represents the observation to interpret, the crosses are the observations generated by Growing Spheres (blue for instances from the same class, black for instances from distinct class). The white cross is the counter example e used to generate explanations.

- Otherwise, there is one or multiple instances from a different class in the sphere.

So far, this seems easily applicable to Anchors since both methods generate instances randomly in an interpretable space around the target instance. If there is no instance in the area of the sphere, Growing Spheres increases the size until it finds one or multiple instances from different class. Once an instance is located within the area of the sphere (hence it can happen at the first iteration), Growing Sphere reduces the area of the sphere until it finds no instance from a different class and returns the position of the closest border.

Thereafter, a feature selection step reduces the dimension of the explanation in order to render it simplify it for a human to understand. This step endeavor is to maximize the sparsity vector that corresponds to the explanation instance. While this instance still from a distinct class, Growing Spheres minimizes the distance of the vector by reducing the feature value difference between the explanation and the target instance for each feature.

Formally, Growing Sphere is a two-step heuristic method minimizing the equation 20 where e^* is the counterfactual example on the border used to explain the target x . A classifier f maps some input space \mathcal{X} of dimension d to an output space $Y = \{1, 1\}$. The final purpose of Growing Spheres is to explain x through the instance $e \in \mathcal{X}$. It computes the norm of the vector $e - x$ based on the l_2 norm to measure the distance between e and x . In order to favor the feature selection step and render the explanation simple to understand for the user, it decides to integrate vector sparsity based on the l_0 norm.

$$e^* = \arg \min_{e \in \mathcal{X}} \{\|x - e\|_2 + \gamma \|x - e\|_0 \mid f(e) \neq f(x)\} \quad (20)$$

As future work, we would like to apply this method to Anchors in order to extend the feature values of the rule to the border. Hence, we consider that these two methods seems related in their way to work and we expect that Growing Spheres may improve Anchors coverage.

8 Conclusion

In this report, we have introduced the most important concepts in interpretable ML and surveyed some recent local works on post-hoc interpretability. We have separated these works into two

distinct categories: Linear explanations and rule based-explanations. Thereafter, we presented further in detail Anchors on two different data types: tabular and textual data. During these five months of internship, we aim at solving the question: “When are anchor based explanation not a good idea?” First, we find out that Anchors is not appropriate when the discretization of the variables is not appropriate. Secondly, we reveal that the replacement strategy does not consider language models.

We noticed the impact of the discretization method on tabular data and argued via an empirical evaluation that MDLP is a better discretization method than the traditional methods provided in Anchor. Afterwards, we presented our new method of interpretability for textual data under the name of pertinent negatives. This novel method explains black box model based on the absence of frequent bi-gram words in a sentence. As future work, we outlined a novel method that seeks out the closest border point. We intend to extend the value of each feature present in the anchor rule via this method. Moreover, we aspire to evaluate MDLP on multi-class models and port our contributions for textual data to other languages. Our experimental results shown that the quality of the anchor depend of the black-box model. Hence, we desire to study if this impact depends of the nature of the complex model *i.e.*, smooth or constant by intervals.

References

- [1] Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. Are emojis predictable? In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers, pages 105–111. Association for Computational Linguistics, 2017.
- [2] Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. SemEval-2018 Task 2: Multilingual Emoji Prediction. In Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, United States, 2018. Association for Computational Linguistics.
- [3] Tiago Botari, Rafael Izbicki, and André C. P. L. F. de Carvalho. Local interpretation methods to machine learning using the domain of the feature space. CoRR, 2019.
- [4] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big Data, 5(2):153–163, 2017.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019.
- [6] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Pai-Shun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen

- Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada, pages 590–601, 2018.
- [7] Finale Doshi-Velez and Been Kim. A roadmap for a rigorous science of interpretability. CoRR, abs/1702.08608, 2017.
 - [8] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In Ruzena Bajcsy, editor, Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993, pages 1022–1029. Morgan Kaufmann, 1993.
 - [9] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. CoRR, abs/1805.10820, 2018.
 - [10] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. ACM Comput. Surv., 51(5):93:1–93:42, 2019.
 - [11] Anil K. Jain and Richard C. Dubes. Algorithms for Clustering Data. Prentice-Hall, 1988.
 - [12] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detryniecki. Comparison-based inverse classification for interpretability in machine learning. In Jesús Medina, Manuel Ojeda-Aciego, José Luis Verdegay Galdeano, David A. Pelta, Inma P. Cabrera, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations - 17th International Conference, IPMU 2018, Cádiz, Spain, June 11-15, 2018, Proceedings, Part I, volume 853 of Communications in Computer and Information Science, pages 100–111. Springer, 2018.
 - [13] Zachary C. Lipton. The mythos of model interpretability. Commun. ACM, 61(10):36–43, 2018.
 - [14] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pages 4765–4774, 2017.
 - [15] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. Artif. Intell., 267:1–38, 2019.
 - [16] Christoph Molnar. Interpretable Machine Learning. 2019. <https://christophm.github.io/interpretable-ml-book/>.
 - [17] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, pages 2060–2069, 2018.

- [18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, page 1135–1144, 2016.
- [19] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), pages 1527–1535, 2018.
- [20] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pages 3145–3153, 2017.
- [21] Robert L. Thorndike. Who belongs in the family? Psychometrika, 18(4):267–276, Dec 1953.