Demonstrating `Quest`: A Query-Driven Framework to Explain Classification Models on Tabular Data

Nadja Geisler
Technical University of Darmstadt
nadja.geisler@cs.tu-darmstadt.de

Benjamin Hättasch
Technical University of Darmstadt
benjamin.haettasch@cs.tu-darmstadt.de

Carsten Binnig
Technical University of Darmstadt
DFKI
carsten.binnig@cs.tu-darmstadt.de

ABSTRACT

Machine learning models are everywhere now; but only few of them are transparent in how they work. To remedy this, local explanations aim to show users how and why learned models produce a certain output for a given input (data sample). However, most existing approaches are oriented around images or text data and, thus, cannot leverage the structure and properties of tabular data. Therefore, we demonstrate `Quest`, a new framework for generating explanations that are a better fit for tabular data. The main idea is to create explanations in the form of relational predicates (called queries hereafter) that approximate the behavior of a classifier around the given sample. For this demo, we use `Quest` on different synthetic and real-world tabular data sets and pair it with a user interface intended to be used during model development by a data scientist working on classification models.

PVLDB Reference Format:
Nadja Geisler, Benjamin Hättasch, and Carsten Binnig. Demonstrating `Quest`: A Query-Driven Framework to Explain Classification Models on Tabular Data. PVLDB, 15(12): 3722 - 3725, 2022.
doi:10.14778/3554821.3554884

1 INTRODUCTION

Machine learning has shown impressive results, surpassing humans in many tasks. However, even data scientists are often unable to explain how or why a result was produced. Yet, this knowledge is important for improving models during development, but also for legal compliance, increasing trust, investigating discrimination and more. Therefore, explainable artificial intelligence (XAI) aims to provide insight on the inner workings of learned models.

Today, different approaches to XAI exist. Local, model-agnostic explanations are particularly interesting: in a nutshell, these approaches try to uncover why a model produced a certain output given a data sample as input (therefore called local) while not using any model internals (therefore called model-agnostic). As an example, think of a learned classifier that is used for monitoring the operation of a wind turbine, illustrated in Figure 1. For this scenario, local and model-agnostic explanations could be used to explain why the learned classifier (independent of its model architecture) indicates for a given data sample (blue dot) whether the operation is safe or should be prohibited. However, many of the

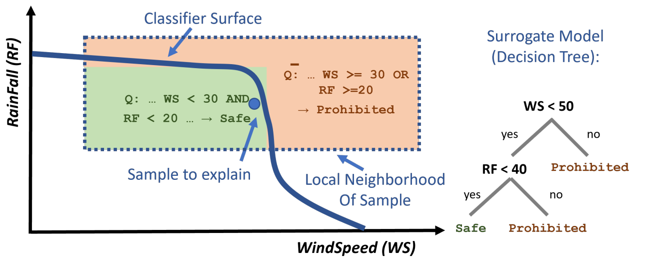
Classifier Surface
RainFall (RF)
$\overline{Q}$: … WS >= 30 OR RF >=20 → Prohibited
Q: … WS < 30 AND RF < 20 … → Safe
Sample to explain
Local Neighborhood Of Sample
WindSpeed (WS)
Surrogate Model (Decision Tree):
WS < 50
yes
no
RF < 40
Prohibited
yes
no
Safe
Prohibited

Figure 1: Simplified example of how queries can explain a classifier output determining if the operation of a wind turbine is safe: The explanation for a given sample is a query pair (Q and $\overline{Q}$) that approximates the classifier in the local neighborhood around the sample. Q explains *why* the model indicated that the operation was *safe*, while $\overline{Q}$ explains *why not* another output was produced by the classifier.

existing approaches for local, model-agnostic explanations target images. For example, (super-)pixels can be colored according to how strongly they influence the model output. Similar approaches can be used for text, where individual tokens are highlighted that cause a model to produce a certain output (e.g., words in a text that caused a sentiment model to label it as "angry"). Some approaches can also be used on tabular data, but they are often too generic for tabular data and do not leverage the structure and properties of the data that might be relevant for explanations. For the example above current approaches might, thus, attribute the output of the classifier (e.g., operation is safe) only to the importance of attributes (e.g., `WindSpeed` or `RainFall`). Alternatively, they might give an example of different conditions under which another output is produced. Instead, what we need are more semantically rich explanations.

Contributions. Therefore, in this paper we demonstrate `Quest`, a new approach for generating explanations that are a better fit for tabular data. The main idea of `Quest` is that explanations are assembled from relational query predicates (called queries in the sequel) that approximate the shape of the separation boundary of a classifier locally. For example, as shown in Figure 1 (left) for the classifier on the before-mentioned wind turbine data, our approach could explain why it indicates the operation is *safe* for the given data sample by producing the query Q: `10 < WindSpeed < 50 AND 15 < RainFall < 40` (shown as green area in Figure 1). The basic idea of how `Quest` creates such query-driven explanations is that it tries to explain the behavior of a classifier in the local neighborhood of a given input sample by training a surrogate model which approximates the shape of the classifier. `Quest` tries to select the

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554884

type of surrogate model with the best fit.The local neighborhood is selected by `Quest` to maximize the area of the neighborhood (i.e., the support of the explanation), while the accuracy of the surrogate model remains high and complexity of the resulting explanation low. As shown in Figure 1 (right) `Quest` can (among other surrogate models) use a decision tree with two labels to approximate the classifier shape in the local neighborhood (dashed border in Figure 1).

Once the surrogate model is trained, `Quest` generates a query pair as an explanation to the user. While Q represents the explanation for *why* the classifier produced a certain output, $\overline{Q}$ provides an explanation of *why not* another output was produced. In Figure 1 (left) Q states that the classifier indicates that operation is safe since the sample was contained in `10 < WindSpeed < 50 AND 15 < RainFall < 40`, while $\overline{Q}$ helps the user understand why the classifier output did not indicate that operation should be prohibited since the sample was not in `10 < WindSpeed < 100 AND 15< RainFall < 60 AND WindSpeed >= 50 OR RainFall >= 40`.

Currently, `Quest` supports several basic surrogate models as explainer templates, including decision tress (as mentioned before) but also linear models ("diagonal" decision boundaries) and a distance-based model ("round" neighborhood). The reason why different explainer templates are supported by `Quest` is so that it can select the surrogate model which best approximates the classifier to be explained in the given neighborhood. Furthermore, `Quest` is designed as a framework where any new explainer templates used to generate query pairs for the explanation can be integrated as long as it can be expressed in queries. This allows `Quest` to be extended in future to support different domains, data sets and classifiers.

To summarize, `Quest` generates local, model-agnostic explanations in the form of queries over tabular data through a new framework approach. These explanations are intuitive and flexible in shape to explain the behavior of a classifier in a local neighborhood around a given data sample. The framework approach allows `Quest` to adapt to the local model behavior and is easily extensible by adding new surrogate models as explainer templates. To enable the community to use and extend the framework, the source code together with the Demo video and further information will be made available publicly at https://link.tuda.systems/quest once the first version is finished.

Outline. The remainder of this demo paper is structured as follows: Section 2 presents some background for explanations on tabular data. Section 3 then gives an overview of the system and a demo scenario for the use by data scientists is outlined in Section 4. Section 5 concludes with a summary and potential avenues of future work.

2 BACKGROUND

In the area of local, model-agnostic explanations that can be used on tabular data the most well known approaches are probably LIME [2] and SHAP [1]. Meanwhile, Anchors [3] is most closely related to our proposed system, since it also generates queries (predicates) as explanations. However, the queries generated by Anchors are much simpler, as discussed below. In the following, we first discuss SHAP and LIME that both explain models based on feature importance.

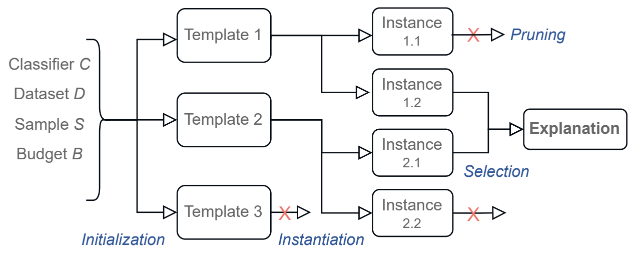

Figure 2: `Quest` Pipeline: As input, `Quest` takes a trained classifier C, a data set D and a data sample S for which the model behavior is explained. The pipeline is composed of multiple steps to select instances of explainer templates (i.e., surrogate models) that approximate C for S. The instance that best approximates C locally produces a query pair as explanation.

Explainers in this category, e.g. LIME and SHAP, provide explanation based on the influence of individual features to the classifier output. Both are also based on local surrogate models. LIME uses a weighted linear model as a surrogate that is trained to approximate the shape of the classifier locally. Based on the weights of the linear model, tabular LIME then derives the feature importance of the different attributes of a table. SHAP, on the other hand, is based on Shapley values: a game theoretic way of quantifying the effect, i.e., the importance, of a feature for a specific model output. As they cannot be computed directly on real-world data, SHAP adapts the surrogate model approach from LIME to estimate the Shapley values for each feature in the sample. As a result, for our example, both SHAP and LIME would produce explanations such as attributing the output *safe* to the `WindSpeed` but they cannot further justify the output, e.g., by producing queries as `Quest` is able to.

However, compared to `Quest`, the resulting queries of anchors are much more limited. First, queries in anchors can only represent queries that describe a fixed "rectangular" shape to approximate the model behavior. For example, in Figure 1 Anchors could actually produce a query that is very similar to Q (green area) as generated by `Quest` to explain *why* the model output indicates that the operation of the wind turbine is safe for the sample. However, depending on the local model behavior, using rectangular shapes—which are in fact conjunctive query predicates—will only work well in some cases while `Quest` supports a much richer set of queries and shapes as we will expand on in Section 3. Second, Anchors does not generate a query $\overline{Q}$ for explaining the *why not* side as `Quest` can do; i.e., Anchors does not explain why a model did not produce the opposite output.

3 SYSTEM OVERVIEW

As discussed before, `Quest` explains the output of a learned classifier for one given sample by providing two areas that are defined by a query pair Q and $\overline{Q}$ as an explanation. In the following, we first discuss the overall pipeline `Quest` uses to generate these query pairs. Afterwards, we explain the different possibilities for the surrogate model (called explainer templates) that can be used to approximate the shape of the classification model we want to explain, before we then discuss the details of how we derive queries as explanations.

3.1 Overall Pipeline

Figure 2 summarizes the process of deriving an explanation through Quest. The process starts with the user providing a data set D, a trained classifier C and a data sample S to explain. Optionally, an accuracy threshold and a budget B that represents the maximum number of predicates that can be used in the query pair Q and $\overline{Q}$ together can be provided. This budget describes a measure of the maximal complexity of the query pair, to ensure it is understandable. With a higher budget, the query pair may be more complex, but it can also explain the behavior of the classifier C more accurately. In the following, we discuss the pipeline phases as shown in Figure 2.

In the first step, suitable explainer templates are selected as surrogate models, depending on the schema of the table. Currently, Quest supports different explainer templates, which we will explain in Section 3.2. However, not all of them can be used for all data sets, e.g., a distance-based surrogate model can only be used on numerical features, not categorical ones. For example, a linear surrogate model that approximates the decision surface of the given classifier C can only be used if the selected features are numerical. During the instantiation phase, Quest creates one or more instances of possible explainer templates (i.e., 1.1 - 2.2 in Figure 2), with their hyperparameters, that should be used to approximate C. This includes the selection of a subset of features for each instance on which the local neighborhood will be defined. Moreover, for each instance of an explainer templates an initial local neighborhood around the sample S is defined which is expanded incrementally. For example, in Figure 1 Quest the explainer template is an instance of decision tree that is used to approximate C in the local neighborhood (i.e., around S). Once a set of instances with different local neighborhoods and surrogate models is created, Quest starts to train the surrogate models using data from D that is included in the selected local neighborhood of the instance. During this phase, pruning eliminates instances early, if they do not seem to be promising candidates. Pruning is based on the size of the neighborhood, complexity of the query and approximation accuracy. The accuracy is determined by classifying data points from the local neighborhood with the surrogate model and the classifier C and evaluating for which fraction they agree. We stop expanding the area of the local neighborhood if the accuracy of the surrogate model is below a given threshold which the user may influence based on their needs. This allows us to find a surrogate model that approximates C with high accuracy while maximizing the area of the local neighborhood (which can be seen as a support for the explanation).

As a last step, the most suitable surrogate model is selected as *explanation* and a query pair is generated as a representation. We select a surrogate model where the accuracy is above a certain threshold, the complexity within the budget and the area of the local neighborhood is maximized. For example, if we have two instances of surrogate models which are above a given accuracy threshold (which can be adapted by the user), Quest selects the one with the largest local neighborhood.

3.2 Explainer Templates

Quest provides different surrogate models that can be used as explainer templates. The idea is that this variety of explainer templates allows Quest to approximate different shapes of C in the local neighborhood. In the following, we briefly explain the currently supported explainer templates. Moreover, since Quest is designed as a framework, new explainer templates can be added. Below, we explain the requirements for new surrogate models to be added.

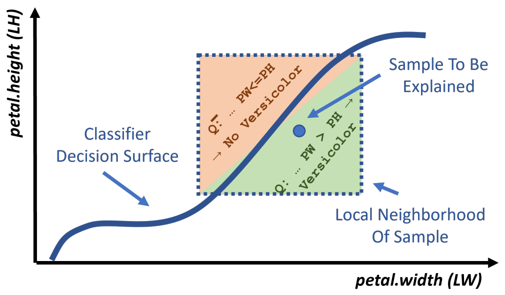
petal.height (LH)
Classifier Decision Surface
$\overline{Q}$: ... PW<=PH → No Versicolor
Q: ... PW > PH → Versicolor
Sample To Be Explained
Local Neighborhood Of Sample
petal.width (LW)

Figure 3: Simplified example explanation using a linear explainer template. It generates a query pair where Q states that `petal.width >= petal.height` causes the classifier output to indicate that the petal came from an Iris Versicolor.

As of now, Quest supports three main explainer templates.

The first template is based on decision trees, that can split the local region recursively into smaller "rectangular" shapes. For example, as shown in Figure 1 this allows the generation of a query pair where $\overline{Q}$ represents a query composed of two rectangular regions (i.e., two different leaves of the decision tree) that are combined in the query through a disjunction. In addition to decision trees, two other explainer templates are supported: one based on linear regression, as well as a distance-based class. The second explainer template approximates the classifier through a linear decision boundary within the neighborhood. This template can be used for explaining classifiers on numerical features that provide comparable metrics. For example, as shown in Figure 3 think of a classifier over a data set of plant petals (similar to the Iris data set) that uses features such as height and width of the petals to classify them (i.e., determine the plant based on the features). The linear explainer could generate a query pair that includes a query Q for the explanation which states since `petal.width >= petal.height` within the neighborhood, the classifier output indicates that the petal came from an Iris Versicolor. Finally, the distance-based explainer class supports classifiers with a "round" shape around the sample. That way, a query explanation can use similarities according to a given distance metric. For example, for the plant data from before, all petals with similar width-to-height ratio could simply be listed as an explanation for why the classifier output for a given sample was produced.

To add a new explainer template to Quest, it needs to fulfill two conditions: The most important is the ability to express their inner workings as a query pair. Additionally, explainer templates need to be able to work within a complexity budget, to ensure that resulting explanations are understandable.

3.3 Queries as Explanations

Based on an instance of a surrogate model, Quest generates a pair of relational queries (Q and $\overline{Q}$) as explanation.

We have chosen queries as the representation of the explanations for several different reasons. First, relational query predicates are extremely expressive while still compact and comprehensible on tabular data. More importantly, using a query pair to explain the local behavior of classification models has the advantage of explaining not only why a model produced an outcome, but also why not.This consideration of query pairs as explanations gives the user an intuitive way of thinking about the local neighborhood, supports generalization on the user's side and keeps the focus on the data instead of, e.g., the surrogate model or feature importance. As such, the combination of Q and $\overline{Q}$ ensures an explanation from both sides of the decision surface. Finally, as mentioned before, the query generation is restricted by a complexity budget B to ensure users are able to understand them well. However, this can be adapted to the target group: Queries can offer much more useful information about the feature space to data scientists specifically, than a single set of feature weights. The simplest way of determining the complexity of queries is the number of predicates, making it easy to compute and comparable. An extension considers a weight for each distinct predicate, based on their perceived complexity. Another way users can think about the query pair is the separation into which area the explanation generalizes to (i.e., the borders of the neighborhood) and a decision surface within that. This is intuitive for data scientists which ensures that they understand the limitations of the explanation and that the explanation itself is interpretable.

4 DEMONSTRATION

In our demonstration, we show how a model developer might use `Quest` to generate explanations that help them infer some of the inner workings of a classifier, compare behavior in regions of the feature space, even across model architectures, and adapt their search with minimal overhead. For the demonstration, we integrate `Quest` into a Jupyter notebook that allows a data scientist to interactively use `Quest` to create explanations and visualize them. We provide several data sets including synthetic and real world data sets, including the adult income data set from the UCI machine learning repository. In the following, we describe the demo scenarios we aim to show in more detail.

Basic Scenario. After loading and preprocessing a data set and training or loading a classifier, the user may specify a data sample for explanation. The resulting explanation is displayed as a pair of readable queries. These query pair is stored in a variable for further use by the data scientist (e.g., to retrieve exemplary data points or data from the same local neighborhood from a different batch).

For the demo, we restrict ourselves to two-dimensional explanations, with regard to the most relevant features, which allows us to provide a visualization for the query pair in which the areas covered by $Q/\overline{Q}$ are colored appropriately as in Figure 1 and Figure 3. Additionally, a variety of evaluation metrics is displayed alongside the query pair: the (normalized) area/coverage of this particular explanation, the accuracy of the decision surface within the local neighborhood, and the class balance between Q and $\overline{Q}$.

Extended Scenario. Because of standardized interfaces in `Quest`, it is possible to explain very different classifiers, as long as they can be specified in a way compatible to scikit-learn's BaseEstimator and the ClassifierMixin. Thanks to the nature of Jupyter notebooks, it is trivial to exchange or add classifiers retroactively.

Furthermore, with an understanding of the internal dataset structure, users may also specify a custom evaluation metric. This can optionally influence the selection of the final explanation from the set of candidates. `Quest` uses the AUC on the development data set by default, but e.g., in the case of sparse data, users might want to calculate this differently, one possibility being newly sampled data labeled by the learned classifier. The visualization may optionally be overlaid with a proportional subset of the development data set, marked in the label assigned by the classifier.

Finally, if a user decides up front to run to create a set of explanations for different samples as well as different classifiers in a batch, the data scientist may specify all necessary input via a YAML configuration file. This will allow them to use a base configuration and vary an arbitrary number of parameters per run.

5 CONCLUSION & FUTURE WORK

In this paper, we demonstrated `Quest`, a framework for query-driven explanations on tabular data. We illustrated why queries make powerful but intuitive representation of local explanations that adapt accurately to the behavior of a complex learned classifier surrounding a given data sample. Additionally, we presented a framework approach to generate these query pairs through a set of explainer templates, each with its own strengths and weaknesses. This makes the framework easy to extend by adding explainer templates. We demonstrated how this framework could be used interactively by data scientists to develop and compare different classifiers, even across model architectures. There are, of course, still various possibilities for improvement. Aside from implementing more complex explainer templates, the most interesting opportunity would result from leveraging the structural information of relational data, in addition to working on singular tables.

ACKNOWLEDGMENTS

This research and development project is/was funded by the German Federal Ministry of Education and Research (BMBF) within the "The Future of Value Creation – Research on Production, Services and Work" program (funding number 02L19C150).

Furthermore, we want to thank hessian.AI as well as DFKI Darmstadt for their support.

REFERENCES

[1] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., Red Hook, NY, USA, 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[2] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[3] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, New Orleans, Louisiana, USA, Article 187, 9 pages.