3.1 Explaining Tuple Non-conformance

Figure 1 shows a screenshot of ExTuNe's graphical user interface, built within a Jupyter Notebook. During the demonstration, we will guide the participants through six steps. We have annotated each step with a circle in Figure 1.

Step ① (Upload reference data): First, the user uploads a reference dataset, whose invariants she is interested to learn. No tuple within the reference dataset should be non-conforming, i.e., the reference data should be clean. For our guided scenario, we upload the tuples with *absence of cardiovascular disease* as the reference data.

Step ② (Learn data invariants): The user issues request for learning data invariants. Following the procedure described in Section 2.1, ExTuNe learns invariants and reports, typically within a few seconds, that the learning is done.

Step ③ (Upload test data): The user uploads the test data containing potentially non-conforming tuples. She wants to identify the non-conforming tuples and understand the cause of non-conformance. For our guided scenario, we use tuples with *presence of cardiovascular disease* as test data.

Step ④ (Top-K non-conforming tuples): The user requests to preview top-K non-conforming tuples from the test dataset. She chooses the value of $K = 10$.

Step ⑤ (Tuple-wise attribute-responsibility heat map): ExTuNe shows top-10 most non-conforming tuples, based on the invariants learned from the reference dataset, in a table where the left-most column denotes identifier of the tuple, followed by the degree of non-conformance (violation). The rest of the table-cells depict a heat map which assigns darker color on more responsible attributes, and lighter color on less responsible ones. The heat map visualizes causes of non-conformance in a tuple-level granularity.

In this dataset, `height` is in cm and `weight` is in kg; `cholesterol` and `glucose` mappings are: 1 = normal, 2 = above normal, 3 = well above normal; `ap_hi` and `ap_lo` correspond to the systolic and diastolic blood pressure measurements, whose normal values are 120 and 80, respectively.

For the first tuple, the non-conformance comes mostly from the abnormally high blood pressures. For the second tuple, besides abnormally high blood pressures, he also has above normal glucose and cholesterol levels. For the sixth tuple, although the blood pressures look normal, she has an abnormally high weight of 180 kg (397 lbs) which is one of the prime causes for her non-conformance.

Step ⑥ (Aggregated attribute responsibility): The user issues a request to visualize attribute responsibility for over-all non-conformance of the test data. ExTuNe visualizes the responsibility of different attributes for non-conformance, aggregated over all tuples in the test data: systolic blood pressure is most responsible for non-conformance, followed by diastolic blood pressure. This is very meaningful since abnormal blood pressure is a primary indicator for cardiovascular disease. This is followed by weight, cholesterol level, and smoking, three other well-known risk factors.

Demonstration engagement. After our guided demonstration, participants will be able to plug their own datasets into ExTuNe. We will also make two additional datasets—MobilePrices[2] (includes attributes such as ram, battery-power, talk-time, etc.) and HousePrices[3] (includes attributes such as area, number-of-rooms, year-built, etc.)—available, as we expect most participants to be familiar with these data domains.

Through the demonstration, we will showcase how ExTuNe can effectively detect non-conforming tuples and explain the causes of the observed non-conformance. Particularly, we expect that the participants will be able to relate the degree of responsibility assigned to different attributes to their real-life experiences (e.g., abnormal blood pressure being a key cause for cardiovascular disease). The key takeaway that our demonstration will highlight is that detecting non-conforming tuples and understanding their causes can significantly help users make decisions about (1) when to trust machine learning models and when not, and (2) how to enrich the training data towards building more robust models.

REFERENCES

[1] A. Fariha, A. Tiwari, A. Radhakrishna, S. Gulwani, and A. Meliou. Data invariants: On trust in data-driven systems. *CoRR*, abs/2003.01289, 2020. http://arxiv.org/abs/2003.01289.
[2] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach: Part 1: Causes. In *UAI*, pages 194–202, 2001.
[3] H. Jiang, B. Kim, M. Y. Guan, and M. R. Gupta. To trust or not to trust A classifier. In *NeurIPS*, pages 5546–5557, 2018.
[4] L. I. Kuncheva and W. J. Faithfull. PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE Trans. Neural Netw. Learning Syst.*, 25(1):69–80, 2014.
[5] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. WHY so? or WHY no? functional causality for explaining query answers. In *MUD@VLDB*, pages 3–17, 2010.
[6] Z. Miao, Q. Zeng, C. Li, B. Glavic, O. Kennedy, and S. Roy. CAPE: explaining outliers by counterbalancing. *PVLDB*, 12(12):1806–1809, 2019.
[7] B. Micenková, R. T. Ng, X. Dang, and I. Assent. Explaining outliers by subspace separability. In *ICDM*, pages 518–527, 2013.
[8] A. A. Qahtan, N. Tang, M. Ouzzani, Y. Cao, and M. Stonebraker. ANMAT: automatic knowledge discovery and error detection through pattern functional dependencies. In *SIGMOD*, pages 1977–1980, 2019.
[9] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.
[10] D. M. J. Tax and K. Müller. Feature extraction for one-class classification. In *ICANN/ICONIP*, pages 342–349, 2003.
[11] M. Tveten. Which principal components are most sensitive to distributional changes? *CoRR*, abs/1905.06318, 2019.
[12] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.

[2] Mobile prices: kaggle.com/iabhishekofficial/mobile-price-classification
[3] House prices: kaggle.com/c/house-prices-advanced-regression-techniques