pysubgroup: Easy-to-Use Subgroup Discovery in Python

Florian Lemmerich[1(✉)] and Martin Becker[2]

[1] RWTH Aachen University, Aachen, Germany
florian.lemmerich@humtec.rwth-aachen.de
[2] University of Würzburg, Würzburg, Germany
becker@informatik.uni-wuerzburg.de

Abstract. This paper introduces the pysubgroup package for subgroup discovery in Python. Subgroup discovery is a well-established data mining task that aims at identifying describable subsets in the data that show an interesting distribution with respect to a certain target concept. The presented package provides an easy-to-use, compact and extensible implementation of state-of-the-art mining algorithms, interestingness measures, and visualizations. Since it builds directly on the established *pandas* data analysis library—a de-facto standard for data science in Python—it seamlessly integrates into preprocessing and exploratory data analysis steps. Code related to this paper is available at: http://florian.lemmerich.net/pysubgroup.

Subgroup discovery [1,5,7] is a data mining method that assumes a population of individuals and a property of these individuals a researcher is specifically interested in. The goal of subgroup discovery is then to discover the subgroups of the population that are statistically "most interesting" with respect to the distributional characteristics of the property of interest, cf. [12]. A typical subgroup discovery result could for example be stated as *"While only 50% of all students passed the exam, 90% of all female students younger than 21 passed."* Here, "female students younger than 21" describes a subgroup, the exam result is the property of interest specified by the user for this task, and the difference in the passing rate is the interesting distributional characteristic. Subgroup discovery identifies such groups in a large set of candidates. Subgroup discovery has been an active research area in our community for more than two decades in order to find more efficient algorithms, improved measures to identify potentially interesting groups, and interactive mining options. It has also been successfully used in many practical applications, see [5] for an overview.

State-of-the-art implementations of subgroup discovery are available in Java (VIKAMINE [2] and Cortana [10]) and R (rsubgroup[1] and SDEFSR[2]). In Python, however, there is only a basic implementation included in the Orange

[1] https://cran.r-project.org/web/packages/rsubgroup/rsubgroup.pdf.
[2] https://cran.r-project.org/web/packages/SDEFSR/vignettes/SDEFSRpackage.pdf.

U. Brefeld et al. (Eds.): ECML PKDD 2018, LNAI 11053, pp. 658–662, 2019.
https://doi.org/10.1007/978-3-030-10997-4_46

workbench.[3] A full featured subgroup discovery implementation that easily integrates with *numpy* and *pandas* libraries, which provide for one of the overall most popular setups for data analysis nowadays, is missing so far. The here presented package *pysubgroup* aims to fill this gap.

1 The pysubgroup Package

The *pysubgroup* package provides a novel implementation of subgroup discovery functions in Python based on the standard *numpy* and *pandas* data analysis libraries. As a design goal, it aims at a concise code base that allows easy access to state-of-the-art subgroup discovery for researchers and practitioners. In terms of algorithms it currently features depth-first-search, an apriori algorithm [6], best-first-search [13], the bsd algorithm [9], and beam search [3]. It includes numerous interestingness measures to score and select subgroups with binary and numeric targets, e.g., weighted relative accuracy, lift, χ^2 measures, (simplified) binomial measures, and extensions to generalization-aware interestingness measures [8]. It also contains specialized methods for post-processing and visualizing results.

Emphasizing usability, subgroup discovery can be performed in just a few lines of intuitive code. Since *pysubgroup* uses the standard pandas DataFrame class as its basic data structure, it is easy to integrate into interactive data exploration and pre-processing with pandas. By defining concise interfaces, *pysubgroup* is also easily extensible and allows for integrating new algorithms and interestingness measures. Based on the Python programming language, *pysubgroup* can be used under Windows, Linux, or macOS. It is 100% open source and available under a permissive Apache license.[4] The source code, documentation and an introductory video is available at http://florian.lemmerich.net/pysubgroup. The package can also be installed via PyPI using `pip install pysubgroup`.

Although *pysubgroup* is currently still in a prototype phase it has already been utilized in practical applications, e.g., for analyzing user motivations in Wikipedia through user surveys and server logs [11].

2 Application Example

Next, we present a basic application example featuring the well-known *titanic* dataset to demonstrate how easy it is to perform subgroup discovery with *pysubgroup*. In this particular example, we will identify subgroups in the data that had a significantly lower chance of survival in the Titanic disaster compared to the average passenger. The complete code required to execute a full subgroup discovery task is the following:

[3] http://kt.ijs.si/petra_kralj/SubgroupDiscovery/.
[4] Other licenses can be requested from the authors if necessary.

```
import pysubgroup as ps
import pandas as pd

data = pd.read_csv("../data/titanic.csv")
target = ps.NominalTarget ('survived', True)
searchspace = ps.createSelectors(data, ignore=['survived'])
task = ps.SubgroupDiscoveryTask (data, target, searchspace,
            resultSetSize=5, depth=2, qf=ps.ChiSquaredQF())
result = ps.BeamSearch().execute(task)
```

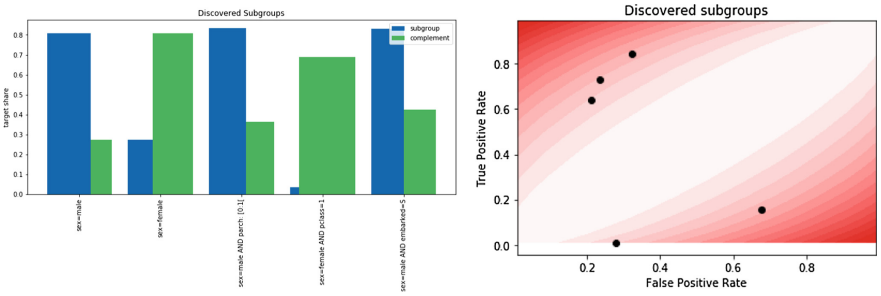

Fig. 1. Visualizations of result subgroups. In the bar visualization on the left, blue bars represent discovered subgroups, green bars their complement in the data. Bar heights indicate the ratio of instances with the property of interest, bar widths show the number of covered instances. On the right, the embedding of the result subgroups in ROC-space is shown. (Color figure online)

The first two lines import the *pandas* data analysis environment and the *pysubgroup* package. The following line loads the data into a standard *pandas* DataFrame object. The next three lines specify a subgroup discovery task. In particular, it defines a target, i.e., the property we are mainly interested in (*'survived'*), the set of basic selectors to build descriptions from (in this case: all), as well as the number of result subgroups returned, the depth of the search (maximum numbers of selectors combined in a subgroup description), and the interestingness measure for candidate scoring (here, the χ^2 measure). The last line executes the defined task by performing a search with an algorithm—in this case beam search. The result is then stored in a list of discovered subgroups associated with their score according to the chosen interestingness measure.

pysubgroup also offers utility functions to inspect and present results. In that direction, the result subgroups and their statistics can be transformed into a separate *pandas* DataFrame that can be resorted, spliced or filtered. Additionally, *pysubgroup* features a visualization component to generate specialized subgroup visualizations with one-line commands, e.g., to create bar visualizations

(cf. Fig. 1a) or to show positions of subgroups in ROC-space [4], i.e., the subgroup statistics in a true positive/false positive space (cf. Fig. 1b). Furthermore, *pysubgroup* enables direct export of results into LaTeX via utility functions. For example, a single function call generates the LaTeX sources for Table 1.

Table 1. Example LaTeX table generated by pysubgroup.

Quality	Subgroup	size_sg	target_share_sg
365.887	sex = male	843	19.1%
365.887	sex = female	466	72.7%
304.403	sex = male ∧ parch: [0:1[	709	16.6%
233.201	sex = female ∧ pclass = 1	144	96.5%
225.957	sex = male ∧ embarked = S	623	17.0%

3 Conclusion

This demo paper introduced the *pysubgroup* package that enables subgroup discovery in a *Python/pandas* data analysis environment. It provides a lightweight, easy-to-use, extensible and freely available implementation of state-of-the-art algorithms, interestingness measures and presentation options.

References

1. Atzmueller, M.: Subgroup discovery. Wiley Interdiscipl. Rev. Data Min. Knowl. Discov. **5**(1), 35–49 (2015)
2. Atzmueller, M., Lemmerich, F.: VIKAMINE – open-source subgroup discovery, pattern mining, and analytics. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012. LNCS (LNAI), vol. 7524, pp. 842–845. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33486-3_60
3. Clark, P., Niblett, T.: The CN2 induction algorithm. Mach. Learn. **3**(4), 261–283 (1989)
4. Flach, P.A.: The geometry of ROC space: understanding machine learning metrics through ROC isometrics. In: International Conference on Machine Learning, pp. 194–201 (2003)
5. Herrera, F., Carmona, C.J., González, P., Del Jesus, M.J.: An overview on subgroup discovery: foundations and applications. Knowl. Inf. Syst. **29**(3), 495–525 (2010)
6. Kavšek, B., Lavrač, N.: APRIORI-SD: adapting association rule learning to subgroup discovery. Appl. Artif. Intell. **20**(7), 543–583 (2006)
7. Klösgen, W.: Explora: a multipattern and multistrategy discovery assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271. American Association for Artificial Intelligence (1996)

8. Lemmerich, F., Becker, M., Puppe, F.: Difference-based estimates for generalization-aware subgroup discovery. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 288–303. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_19
9. Lemmerich, F., Rohlfs, M., Atzmueller, M.: Fast discovery of relevant subgroup patterns. In: International Florida Artificial Intelligence Research Society Conference (FLAIRS), pp. 428–433 (2010)
10. Meeng, M., Knobbe, A.: Flexible enrichment with Cortana-software demo. In: Proceedings of BeneLearn, pp. 117–119 (2011)
11. Singer, P., et al.: Why we read Wikipedia. In: International Conference on World Wide Web (WWW), pp. 1591–1600 (2017)
12. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Zytkow, J. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63223-9_108
13. Zimmermann, A., De Raedt, L.: Cluster-grouping: from subgroup discovery to clustering. Mach. Learn. **77**(1), 125–159 (2009)