Breaking Down Accuracy with Subspace Optimization

Donatella Firmani
Sapienza University
Rome, Italy
donatella.firmani@uniroma1.it

Giorgio Grani
Sapienza University
Rome, Italy
g.grani@uniroma1.it

Flavia Tagliafierro
Sapienza University
Rome, Italy
flavia.tagliafierro@uniroma1.it

ABSTRACT

This paper introduces a Python toolkit to break down accuracy of classification models, by identifying high-accuracy portions of the test dataset and thus facilitating a deeper understanding of where the model performs best and where it falls short.

Given a dataset, a classification model and an accuracy threshold, our system returns a *range query* that selects the largest sub-space of the test dataset on which the classifier achieves higher accuracy. The toolkit allows users to interactively explore such sub-space, by adjusting the returned ranges with graphical elements and observing the change in the model's accuracy and data distributions. Ranges can be manually initialized to highlight the strengths and weaknesses of the model in different scenarios.

The core of our method consists of a mixed-integer optimization algorithm. Demonstration on real-world datasets and a selection of models show that our toolkit can serve as an effective way to understand performance across different data segments.

1 INTRODUCTION

Classification models have become ubiquitous in a wide range of applications, from healthcare to finance. These models, powered by sophisticated algorithms, can play a pivotal role in decision-making processes. The standard de-facto metric for assessing model performances is *global accuracy*, that is, the fraction of correctly classified data points on a test set. However accuracy can vary wildly depending on the characteristics of a certain dataset, and there are two main reasons for which global accuracy alone can be misleading:

- *High global accuracy ≠ good model.* High-accuracy models can exhibit bias, performing well for over-represented groups or types of data, but not for others that are equally important.
- *Modest global accuracy ≠ bad model.* Different applications can tolerate errors in sub-spaces of the data that are not critical.

Intuitively, breaking down global accuracy would build confidence in using the model for real-world decisions. Knowing where the model excels and where it does not would allows users to make informed decisions about its applicability and limits, guiding data collection, feature engineering, and model tuning.

Unfortunately, to the best of our knowledge, there are no standard approaches to perform accuracy break down. In this paper, we initiate this study by demonstrating our current solution, which is based on a simple - yet effective - break-down policy.

Our break-down policy is illustrated in Figure 1. In the figure, each point represents a different instance of a two-dimensional test set and the global accuracy is $\frac{25-6}{25} = 0.76$. Given a threshold such as 0.9 to quantify high accuracy, our tool can compute a

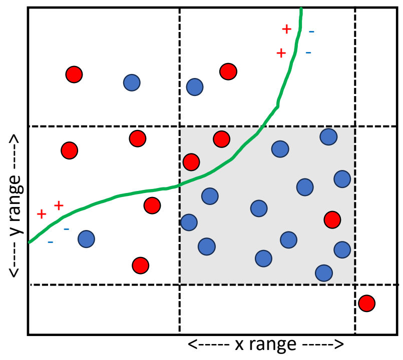


Figure 1: High-accuracy subspace in two dimensions. The space is separated in two regions according to the green line. Data points to the left are classified as red (+) and data points to the right are classified as blue (-). The ground-truth class of each instance is represented by the color of each point.

partition of the evaluation's scope in two parts: a high-accuracy sub-space, in grey, and a low-accuracy sub-space, in white. Accuracy of the two areas is respectively $accuracy(grey) = \frac{14}{15} \approx 0.93$, above the threshold, and $accuracy(white) = \frac{5}{10} = 0.5$.

In order to compute the sub-space partitioning, we design a mixed integer optimization problem and use a state-of-the-art optimizer [4] to find the hyper-rectangular sub-space that has maximum size and accuracy above threshold. The size of a sub-space is simply defined as the number of enclosed data points.

The computed result is returned to the user as a *range query*, which provides a compact and human readable way to identify high-accuracy data points and low-accuracy data points as respectively those inside and outside the ranges.

Contribution. We call our tool `SCOPE`, standing for **S**ubspace **C**lassification **OP**timization and **E**xplanation. We slightly abuse the term "explanation" because computing high-accuracy ranges can be broadly thought as an attempt to interpret a model's classification patterns. Users of the demo will be able to use our tool either standalone or together with the SHAP [5] explanation tool to show the feature importance in the high-accuracy area.

`SCOPE` is available on GitHub[1]. The high-level architecture of the system consists of four modules:

(1) *Dataset module.* A module to load/select datasets and features. We provide a collection of pre-loaded datasets. Users can bring their own datasets by uploading files in a dedicated folder.
(2) *Classification module.* A module to select/train a classification model, make predictions and compute accuracy. We include by default three classification models, dubbed Random Forests, Logistic Regression, and feedforward Neural Network. Users can bring their own models by implementing a Python class.

[1]https://github.com/Flaviatagliafierro/Scope

(3) *Optimization module.* A module that takes as input the dataset, the features and the classification model as described in the previous modules. Then, it solves a mixed integer optimization problem to finds the largest sub-space with a certain minimum level of accuracy desired. The module adopts GUROBI [4] as an optimizer and returns a range query that partitions the test data space into an high-accuracy and a low-accuracy sub-space.
(4) *Display module.* A graphical interface to interact with the automatically computed ranges, adjust them manually and observe changes in the accuracy and feature importance.

The existing literature is rich with applications where high-accuracy classification models deviate from the desired behaviour in unexpected ways. One popular example is a classifier for images of huskies and wolves [7], unable to recognize pictures of huskies without snow in the background. Another example is a 95% accuracy Entity Matching model unable to recognize two different songs with the same duration [3]. Range queries are a popular tool in data analysis. We mention the recent work [8] where ranges are automatically computed to identify *fair* sub-spaces, according to different fairness constraints [6].

Overview. In Section 2 we formalize the sub-space optimization problem at the heart of our approach. In Section 3 we illustrate different demo scenario using publicly available datasets. Finally Section 4 provides conclusive remarks.

2 SUBSPACE OPTIMIZATION PROBLEM

We define a Subspace Optimization Problem (SOP) formulated as a *mixed-integer mathematical program* as the problem of finding the subspace containing the larger number of points and respecting a minimum level of accuracy, decided by the user.

SOP takes as input the threshold accuracy β and returns lower and upper bounds for each feature (l^k and u^k for the k-th feature) representing a **range query** for identifying a high-accuracy subspace. Intuitively, the fact that there is a subspace with higher performances means two things: on the one hand, we can be more confident of the results if they lay in the subspace, and on the other hand, we may show the model is unbalanced for certain regions. In the rest of this section we describe the main components of the problem: sets, decision variables, parameters, constraints and objective function.

Sets. The main sets are:

- $I = \{1, \dots, P\}$ represents the set of indices of the points in the dataset.
- $(x_i, y_i)_{i \in I}$ represents the dataset, where $y_i \in \mathbb{R}$ is the label, $x_i \in \mathbb{R}^K$ is the vector of features.

The number of features in the dataset is given by the integer $K > 0$, and x_i^k is the k-th feature of x_i.

Decision variables. The decision variables are:

- v_i, $i \in I$, are binary indicator variables indicating the instances to be included in the final ranged problem. We use the notation v for the vector of all the variables v_i. The solution will have $v_i = 1$ if the i-th point belongs to the optimal subspace, and $v_i = 0$ otherwise.
- p_i^k, $i \in I$ and $k = 1, \dots, K$, are binary indicator variables indicating if the i-th instance belongs inside the k-th range. The solution will have $p_i^k = 1$ if the i-th point belongs to the k-th optimal range, and $p_i^k = 0$ otherwise.
- s_i^k, $i \in I$ and $k = 1, \dots, K$, are auxiliary binary indicator variables indicating to which disjuction the i-th instance belongs when it is not inside the k-th range.
- $u^k \in \mathbb{R}$ for $k = 1, \dots, K$, are continuous variables representing the upper bounds of the ranges. We indicate with u the vector of all the variables u^k.
- $\ell^k \in \mathbb{R}$, $\ell^k \leq u^k$ for $k = 1, \dots, K$, are continuous variables representing the lower bounds of the ranges. We indicate with ℓ the vector of all the variables ℓ^k.

Parameters and additional sets. In the following, we describe the main parameters and additional sets:

- $\beta \in (0, 1]$ is the threshold value for the accuracy. It is decided interactively by the user. Increasing β will lead to progressively higher accuracy of the optimal subspace found.
- r_k for $k = 1, \dots, K$ are the size of the ranges in the entire dataset.
- c_i, for $i \in I$, indicates if the label returned by the classification model is correct or not. In other words, $c_i = 1$ is the model ϕ correctly classifies the i-th vector x_i, i.e. $c_i = 1$ if $\psi(x_i) = y_i$, and $c_i = 0$ otherwise.
- $M^k = 1 + \max_{i \in I} \{|x_i|\}$, for $k = 1, \dots, K$ are the big-M values used to activate the disjunctive constraints for the lower and upper bounds following the methodology in [1].
- $K_\ell^a \subseteq \{1, \dots, K\}$ is the set for which there exists an upper bound a_ℓ^k on ℓ^k specified by the user
- $K_\ell^b \subseteq \{1, \dots, K\}$ is the set for which there exists a lower bound b_ℓ^k on u^k specified by the user.
- $\hat{K}_\ell^a \subseteq \{1, \dots, K\}$ is the set for which there exists a fixed value $\hat{a}_\ell^k$ for ℓ^k decided by the user.
- $\hat{K}_\ell^b \subseteq \{1, \dots, K\}$ is the set for which there exists a fixed value $\hat{b}_\ell^k$ for u^k decided by the user.
- $\epsilon > 0$ is a small value.

Constraints and objective function. Figure 2 formally defines SOP(β) as the problem of finding the ranges of the largest subset of instances having an accuracy $\geq \beta$.

The high-level description of the main constraints follows.

- (2) indicates that the accuracy of the subspace should not be lower than β.
- (3) − (4) − (5) − (6) are disjunctive constraints indicating that the k-th component of the i-th point, x_i^k, belongs to the interval $l^k \leq x_i^k \leq u^k$ or it is outside, i.e. $x_i^k < l^k$ or $x_i^k > u^k$.
- (7) are constraints indicating that if the i-th point is selected then it must belong to all the K ranges, i.e. $v_i = 1 \iff p_i^k = 1$ for $k = 1, \dots, K$.
- (8) are constraints indicating that if a point belongs to all the K ranges then it has to be selected, i.e. if $p_i^k = 1$ for $k = 1, \dots, K$ then $v_i = 1$.
- (9) − (10) are bound constraints limiting the choice of the ranges below the levels specified externally.
- (11) − (12) are fixing constrains, blocking the ranges to the levels specified externally.

The objective function (1) is obtained according to the results in [2], as a transformation by scalarization of the lexicographic optimization problem $lex \max_{v,\ell,u} \left(\sum_{i \in I} v_i, -\frac{1}{K+\epsilon} \sum_{k=1}^{K} \frac{u_k - l_k}{r_k} \right)$ by noticing that the first objective is a γ−function.

$$\mathsf{SOP}(\beta) = \max_{v,\ell,u} \sum_{i\in I} v_i - \tfrac{1}{K+\epsilon} \sum_{k=1}^{K} \tfrac{u_k - l_k}{r_k} \tag{1}$$

$$\text{s.t. } \sum_{i\in I} c_i v_i - \beta \sum_{i\in I} v_i \geq 0 \tag{2}$$

$$-M^k(1-p_i^k) + \ell^k - x_i^k p_i^k \leq 0, \ \ \forall\, i \in I, k = 1,\dots,K \tag{3}$$

$$x_i^k p_i^k - u^k - M^k(1-p_i^k) \leq 0, \ \ \forall\, i \in I, k = 1,\dots,K \tag{4}$$

$$M^k(p_i^k + 1 - s_i^k) - \epsilon(1-p_i^k) + \ell^k \geq x_i^k, \forall\, i \in I, k = 1,\dots,K \tag{5}$$

$$-M^k(p_i^k + s_i^k) + \epsilon(1-p_i^k) + \ell^k \leq x_i^k, \ \ \forall\, i \in I, k = 1,\dots,K \tag{6}$$

$$K v_i - \sum_{k=1}^{K} p_i^k \leq 0, \ \ \forall\, i \in I \tag{7}$$

$$v_i - \sum_{k=1}^{K} p_i^k \geq 1 - K, \ \ \forall\, i \in I \tag{8}$$

$$\ell^k \geq a_\ell^k, \quad k \in K_\ell^a \tag{9}$$

$$u^k \leq b_u^k, \quad k \in K_\ell^b \tag{10}$$

$$\ell^k = \hat{a}_\ell^k, \quad k \in \hat{K}_\ell^a \tag{11}$$

$$u^k = \hat{b}_u^k, \quad k \in \hat{K}_\ell^b \tag{12}$$

Figure 2: Definition of the Subspace Optimization Problem for finding the largest subset of instances with accuracy $\geq \beta$

Implementation details. We solve SOP with the Gurobi Optimizer Python library with academic free licence [4]. The optimizer eventually returns the optimal solution.[2] We also include a parametric time budget, which by default is set to 3 minutes, after which the current best solution is returned.

3 DEMONSTRATION SCENARIOS

In Figure 3, we illustrate the user interface with a simple dataset from Kaggle, dubbed `Titanic` [3]. The dataset contains 889 records representing Titanic passengers and their survival status. Features include the traveling class ("PClass"), the cost of their ticket ("Fare"), demographics data such as "Sex" and "Age", the size of the group, divided into siblings/spouses ("SibSp") and parents/children ("ParCh"). For people traveling alone and we have that "SibSp"+"ParCh"=0.

Initial set-up. In step ① the user can choose a dataset and specify a target variable for the classification task, e.g., "Survived". Upon a new selection, features are shown below and the user is able to select them and inspect the corresponding distribution of the datapoints in the test set, split by the "Survived" class.

Users can bring their own datasets by uploading two files in a dedicated folder: a CSV file with all the instances and a metadata file with the type of each feature among `natural-ordering`, `ordinal-feature` and `no-ordering`. Examples of features of the first type are numerical features, although there can be textual features that make sense to sort lexicographically. Features of the second type include categorical features that support a domain-based ordering relation, such as "Degree". Features of the third type include features that have no natural or domain ordering such as "Country". The latter type, i.e., `no-ordering`, cannot be added to the optimization problem but can still be used here, by the classifier and the explanation tool if the user wants to.

Users can optionally upload a third file to specify how to split the dataset into training and test. If such a file is not provided, SCOPE does a random 70% – 30% split.

Classification results. In step ② the user can choose a classification model. We provide three options for the demonstration: Random Forest, Logistic Regression, and Feedforward Neural Network. Upon a new selection, that is, "Random Forest" in Figure 3, the tool trains a new classification model and then generates a classification results report. The user may customize the report choosing to show a combination of accuracy ("Accuracy only"), confusion matrix ("Full report"), and distribution of the feature importance values. By default, the latter is computed with the popular SHAP [5] explanation toolkit ("Shap feature importance"). Users cannot bring new models during the live demo. However, they can bring their own models when using SCOPE off-line, by implementing a class in a dedicated Python module. Pre-trained models are also supported.

Manipulating ranges. In ③, the user can interact with range sliders to decide to set up bounds that will be included in the optimization module. Below the ranges, a textbox displays the size of the selected sub-space as a percentage. In Figure 3, we show the initial configuration of the sliders, with the textbox showing the value 100, indicating the entire test set will be included.

Ranges can be manually adjusted (step ④) by simply moving the sliders. Upon adjustments, the value of the selected percentage of the test set changes, and the results in ① and ② are updated by restricting the scope to the selected range. For instance, by restricting the "Age" range to 18-35, plot in ① would show the "PClass" distribution by considering only data points of the test set that have an age between 18 and 35 years. Analogously, the accuracy in ② would become the accuracy of the selected sub-space.

The button "Reset Filters" in part ⑤ resets all the ranges.

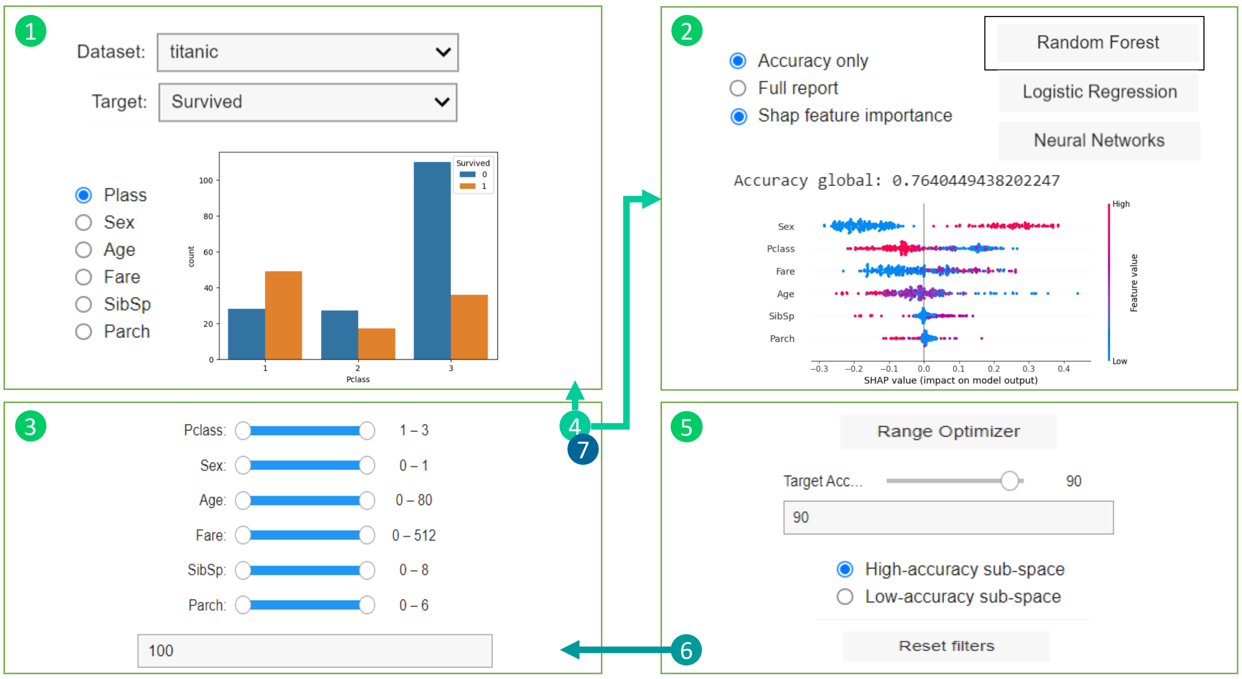
Sub-space optimization. In step ⑤, the user can set the accuracy threshold and run the optimization module to solve the SOP problem defined in Section 2, initialized with the bounds in ③. As soon as optimal ranges are ready, the slider buttons are updated (step ⑥) and therefore the visualizations in ① and ② (step ⑦, analogously to step ④). With the setup in Figure 3, that is no initial range restriction and target accuracy equals to 90%, the solution found has size 23% of the test set and corresponds to "Pclass"=$[2,3]$, "Age"=$[17,30]$ and "Fare"=$[7,13]$. The accuracy achieved on this subspace is 0.90625, just above the threshold. Such subspace can be intuitively described as young passengers with cheap tickets, travelling in second and third class.

During the live demo, users will be able to inspect the full report in ② (omitted from the figure for space constraints) and discover that most of those passengers did survive[4] and that even

[2]https://pypi.org/project/gurobipy/

[3]https://www.kaggle.com/competitions/titanic

[4]This might be surprising for those remembering the *Titanic* movie by James Cameron.


Dataset: titanic

Target: Survived

[✓ Plass] Sex Age Fare SibSp Parch

[✓ Accuracy only] Full report [✓ Shap feature importance]

Random Forest | Logistic Regression | Neural Networks

```
Accuracy global: 0.7640449438202247
```

Pclass:	1 – 3
Sex:	0 – 1
Age:	0 – 80
Fare:	0 – 512
SibSp:	0 – 8
Parch:	0 – 6

100

Range Optimizer

Target Acc... 90

90

[✓ High-accuracy sub-space] Low-accuracy sub-space

Reset filters

Figure 3: Main demonstration functionalities.

a simple classifier was able to learn this pattern with exceptional accuracy.

Other scenarios. During the live demo, users can change dataset, classification model, lower and upper ranges, try different accuracy thresholds are run the optimizer with different settings. They will be able to highlight strengths and weaknesses of selected classifiers and learn more about the selected dataset. Finally, all the visualizations concern by default the high-accuracy sub-space found by the optimizer. However, the user can select the "Low-accuracy sub-space" option in part ⑤ and inspect the part outside the ranges.

4 CONCLUSIONS

This paper presented SCOPE, a Python toolkit to enhance the evaluation of classification models by providing a more granular analysis of model performance. Our approach diverges from the traditional reliance on global accuracy and represents a significant improvement over manual inspection of accuracy variation, thanks to a mixed integer optimization algorithm for automatic identification of high-accuracy sub-spaces within the test set.

During the live demo, users will be able to explore these sub-spaces by running the optimization algorithm and inspecting results. Furthermore, they will be able to bring their own datasets and uncover common characteristics among data points that are correctly or incorrectly classified, thereby narrowing the scope for in-depth examination of classification patterns.

Intuitive graphical elements such as slider buttons make the toolkit accessible to a diverse range of users, regardless of their technical expertise, enabling them to gain a deeper understanding of how changes in data can impact model accuracy and to learn to identify possible biases that affect performance variations.

Future works. Future works will focus on expanding the toolkit with other performance metrics like precision and recall and adding constraints to solution sizes, further increasing its applicability. Other envisioned expansions include compact representations alternatives to hyper-rectangles, such as cluster of data points which are correctly / incorrectly labeled. Finally we plan to explore efficient methods to compute approximate solutions over large datasets.

ACKNOWLEDGEMENTS

The authors have been partially supported by the following research grants: SEED PNR Project "FLOWER" "Frontiers in Linking records: knOWledge graphs, Explainability and tempoRal data", Sapienza Research Project B83C22007180001 "Trustworthy Technologies for Augmenting Knowledge Graphs" and HORIZON Research and Innovation Action 101135576 INTEND "Intent-based data operation in the computing continuum".

REFERENCES

[1] Dimitris Bertsimas and Robert Weismantel. 2005. (2005).
[2] Marianna De Santis, Giorgio Grani, and Laura Palagi. 2020. Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for biobjective integer programming. *European Journal of Operational Research* 283, 1 (2020), 57–69.
[3] Vincenzo Di Cicco, Donatella Firmani, Nick Koudas, Paolo Merialdo, and Divesh Srivastava. 2019. Interpreting deep learning models for entity resolution: an experience report using LIME. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management @ SIGMOD.*
[4] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. https://www.gurobi.com
[5] 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., 4765–4774.
[6] Arvind Narayanan. 2018. Translation tutorial: 21 fairness definitions and their politics. In *Proc. conf. fairness accountability transp., new york, usa*, Vol. 1170. 3.
[7] 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*. 1135–1144.
[8] Suraj Shetiya, Ian P Swift, Abolfazl Asudeh, and Gautam Das. 2022. Fairness-aware range queries for selecting unbiased data. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 1423–1436.