3.1 Introduction to the Hands-on Resources

- Test notebook.
- Data sets and black boxes notebook.
- Interpretable representation notebook.
- Data sampling notebook.
- Explanation generation notebook.
- Slack communication.

Alex Hepburn

Test Notebook

https://github.com/fat-forensics/events/blob/master/resources/2020_ecml-pkdd/notebooks/0-environment-test.ipynb

Testing the Tutorial Environment

This notebook is a pruned version of "how to build LIME yourself" *How-To Guide* taken from the FAT Forensics documentation.

Regardless of whether you are using *Colab*, *Binder* or running the tutorial materials directly on your personal computer, you should execute this notebook to check whether everything functions correctly.

We will discuss all of these steps in more detail during the hands-on session in Part 3 of the tutorial.

> **For now, you do not need to worry about the meaning of any of these. Just make sure it runs.**

First, let's set up required packages and fix the random seed for reproducibility.

Data Sets Notebook

https://github.com/fat-forensics/events/blob/master/resources/2020_ecml-pkdd/notebooks/1-data-sets.ipynb

Introducing Data Sets and Black Boxes

This notebook briefly introduces example data sets that we will use throughout the tutorial:

- *Two Moons* -- a two-dimensional toy data set; and
- *Bikes Sharing* -- a real-life data set that captures bike rentals (available via the UCI repository).

Moreover, it discusses two classifiers that will serve as our *black boxes* (trained on said data):

- Random Forest -- utilises the `sklearn.ensemble.RandomForestClassifier` class and serves as a probabilistic black box; and
- Support Vector Machine -- uses the `sklearn.svm.SVC` class and serves as a non-probabilistic black box.

Reproducibility To ensure reproducibility of the data -- random generation of Two Moons and train/test split of both data sets -- we encourage fixing the random seeds of the Python's and numpy's `random` modules. This can be easily done by calling `fatf.setup_random_seed(42)`, where `42` is the chosen random seed. This step is already integrated into the data set creation functions -- `fatf_ecml.generate_2d_moons` and `fatf_ecml.generate_bikes` -- and can be accessed with their `random_seed` parameter, e.g.:

```
>>> fatf_ecml.generate_2d_moons(random_seed=42)
```

If called without this parameter or with `None`, the random seed is not fixed.

Similarly, you may wish to train reproducible models. This can be achieved by using the `random_state` parameter, which every scikit-learn model class has, e.g., `sklearn.svm.SVC(random_state=42)`. Again, this functionality has already been integrated into the model creation functions in the `fatf_ecml` library -- `fatf_ecml.get_random_forest` and `fatf_ecml.get_svc` -- and can be accessed via their `random_seed` parameter, e.g.:

Interpretable Representation Notebook

https://github.com/fat-forensics/events/blob/master/resources/2020_ecml-pkdd/notebooks/2-interpretable-representations.ipynb

Binary Interpretable Representations for Tabular Data

In search of a binary interpretable representation for tabular data.

Toy Example -- Two Moons

Loading the Data

```
[5]  moons_X, moons_X_test, moons_y, moons_y_test = fatf_ecml.generate_2d_moons(
         random_seed=42)
```

```
20-Sep-01 16:05:31 fatf         INFO     Seeding RNGs using the input parameter.
20-Sep-01 16:05:31 fatf         INFO     Seeding RNGs with 42.
```

Data Sampling Notebook

https://github.com/fat-forensics/events/blob/master/resources/2020_ecml-pkdd/notebooks/3-data-sampling.ipynb

Data Sampling

This notebook goes over a selection of data sampling methods and investigates their behaviour with respect to the produced data. To this end, we use implementations included in the `fatf.utils.data.augmentation` module. In particular, we will look at four different sampling methods for tabular data:

- Normal Sampling;
- Truncated Normal Sampling;
- Mixup; and
- Normal Class Discovery.

These four approaches are implemented in FAT Forensics and are ready for you to use out of the box. Nonetheless, writing your own sampler is relatively easy. You can use the abstract `fatf.data.utils.augmentation.Augmentation` class as the foundation of your own method. It implements a collection of helper methods and in most of the cases writing your own `__init__` and `sample` methods suffices.

Toy Example -- Two Moons

Let's start with the Two Moons data set -- it is two-dimensional, which allows us to visualise the behaviour of each sampler and more easily convey the idea behind them.

Explanation Generation Notebook

https://github.com/fat-forensics/events/blob/master/resources/2020_ecml-pkdd/notebooks/4-explanation-generation.ipynb

Explanation Generation

This notebook combines insights from the *Interpretable Representations* and *Data Sampling* exercises with the *Explanation Generation* step to build local surrogate models. It shows how to *weight data* and *achieve explanation sparsity* when composing explanations of black-box predictions. In particular, we discuss:

- a range of distance functions;
- kernels for transforming distances into similarity scores;
- (interpretable) feature selection mechanisms to induce explanation sparsity;
- choice of surrogate model types; and
- types of explanations that can be generated, their meaning and interpretation.

We introduce these procedures for the *Two Moons* data set, which helps us to convey the core concepts with visualisations. Then, we move on to the more complicated example with the bikes sharing data set.

Toy Example -- Two Moons

Slack

https://fatforensicsevents.slack.com/

(Registration via a separate URL.)

Next Up

Active Participation

https://github.com/fat-forensics/events/tree/master/resources/2020_ecml-pkdd/notebooks/

(Alex Hepburn & Kacper Sokol)

Next Next Up

Summary and Farewell

(Raul Santos-Rodriguez)