9. Model persistence

After training a scikit-learn model, it is desirable to have a way to persist the model for future use without having to retrain. The following sections give you some hints on how to persist a scikit-learn model.

9.1. Python specific serialization

It is possible to save a model in scikit-learn by using Python's built-in persistence model, namely pickle:

```
>>> from sklearn import svm
>>> from sklearn import datasets
>>> clf = svm.SVC()
>>> X, y= datasets.load_iris(return_X_y=True)
>>> clf.fit(X, y)
SVC()

>>> import pickle
>>> s = pickle.dumps(clf)
>>> clf2 = pickle.loads(s)
>>> clf2.predict(X[0:1])
array([0])
>>> y[0]
0
```

In the specific case of scikit-learn, it may be better to use joblib's replacement of pickle (`dump` & `load`), which is more efficient on objects that carry large numpy arrays internally as is often the case for fitted scikit-learn estimators, but can only pickle to the disk and not to a string:

```
>>> from joblib import dump, load
>>> dump(clf, 'filename.joblib')
```

Later you can load back the pickled model (possibly in another Python process) with:

```
>>> clf = load('filename.joblib')
```

Note: `dump` and `load` functions also accept file-like object instead of filenames. More information on data persistence with Joblib is available here.

9.1.1. Security & maintainability limitations

pickle (and joblib by extension), has some issues regarding maintainability and security. Because of this,

- Never unpickle untrusted data as it could lead to malicious code being executed upon loading.
- While models saved using one version of scikit-learn might load in other versions, this is entirely unsupported and inadvisable. It should also be kept in mind that operations performed on such data could give different and unexpected results.

In order to rebuild a similar model with future versions of scikit-learn, additional metadata should be saved along the pickled model:

- The training data, e.g. a reference to an immutable snapshot
- The python source code used to generate the model
- The versions of scikit-learn and its dependencies
- The cross validation score obtained on the training data

This should make it possible to check that the cross-validation score is in the same range as before.

Aside for a few exceptions, pickled models should be portable across architectures assuming the same versions of dependencies and Python are used. If you encounter an estimator that is not portable please open an issue on GitHub. Pickled models are often deployed in production using containers, like Docker, in order to freeze the environment and dependencies.

If you want to know more about these issues and explore other possible serialization methods, please refer to this talk by Alex Gaynor.

9.1.2. A more secure format: `skops`

skops provides a more secure format via the **`skops.io`** module. It avoids using **`pickle`** and only loads files which have types and references to functions which are trusted either by default or by the user. The API is very similar to `pickle`, and you can persist your models as explain in the docs using **`skops.io.dump`** and **`skops.io.dumps`**:

Toggle Menu

```
import skops.io as sio
obj = sio.dumps(clf)
```

And you can load them back using `skops.io.load` and `skops.io.loads`. However, you need to specify the types which are trusted by you. You can get existing unknown types in a dumped object / file using `skops.io.get_untrusted_types`, and after checking its contents, pass it to the load function:

```
unknown_types = sio.get_untrusted_types(obj)
clf = sio.loads(obj, trusted=unknown_types)
```

If you trust the source of the file / object, you can pass `trusted=True`:

```
clf = sio.loads(obj, trusted=True)
```

Please report issues and feature requests related to this format on the skops issue tracker.

9.2. Interoperable formats

For reproducibility and quality control needs, when different architectures and environments should be taken into account, exporting the model in Open Neural Network Exchange format or Predictive Model Markup Language (PMML) format might be a better approach than using `pickle` alone. These are helpful where you may want to use your model for prediction in a different environment from where the model was trained.

ONNX is a binary serialization of the model. It has been developed to improve the usability of the interoperable representation of data models. It aims to facilitate the conversion of the data models between different machine learning frameworks, and to improve their portability on different computing architectures. More details are available from the ONNX tutorial. To convert scikit-learn model to ONNX a specific tool sklearn-onnx has been developed.

PMML is an implementation of the XML document standard defined to represent data models together with the data used to generate them. Being human and machine readable, PMML is a good option for model validation on different platforms and long term archiving. On the other hand, as XML in general, its verbosity does not help in production when performance is critical. To convert scikit-learn model to PMML you can use for example sklearn2pmml distributed under the Affero GPLv3 license.

Toggle Menu