**Published on Medium: https://medium.com/@spettiett/how-to-pickle-your-trained-model-f4b7051babaa**

**How to Pickle your Trained Model**



As a data scientist working on machine learning problems you realize that these models can take a longtime to run. So, once your model is trained is there a way to save it and use the saved model to make predictions without re-running the training process over again?

The answer is, **YES**, and the solution is “*Pickle*” …

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What is Python Pickle:**

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. The pickling process simply converts an object to a stream of bytes. This stream can then be reconverted to the original object later.

**Pros:**

* Storing results in a data file for later use, can save processing time on a python object
* Can serialize most python objects
* Part of the Standard Library — no external library needed
* Easy to code

**Cons:**

* ONLY supported by Python
* Needs same python version to work correctly between pickling and unpickling objects
* The pickle module is NOT secure for opening pickled messages from an untrusted user

(See “Warning Message” listed below)

**What can be pickled and unpickled?**

The following types can be pickled:

* integers, floating point numbers, complex numbers
* strings, bytes, bytearrays
* tuples, lists, sets, and dictionaries containing only pickable objects
* functions defined at the top level of a module (using [def](https://docs.python.org/3/reference/compound_stmts.html#def), not [lambda](https://docs.python.org/3/reference/expressions.html#lambda))
* built-in functions defined at the top level of a module
* classes that are defined at the top level of a module

Which includes, pickling and unpickling a “*trained”* object. Let’s look at the steps below that demonstrate how this is accomplished.

**Steps | “Pickling” and “Unpickling” a Trained Model**

**How to pickle an object?**

**Python Code:**

# Import library

import pickle # To use any of the pickling methods, import library.

# Build Training Model

rf\_clf = RandomForestClassifier()

model = rf\_clf.fit(X\_train, y\_train)

# Pickle Training Object

save\_clf = open('./RandomForestClf.pickle', 'wb') # Open data file and write bytes.

pickle.dump(model, save\_clf) # To store the result in a file (Pickle).

save\_clf.close()

# **NOTE:**

Pickling converted our object to a stream of bytes. So, it shouldn't be a surprise that it's hard to read.

**How to get the data back to its original form?**

**Python Code:**

# Unpickle Training Object

load\_clf = open('./RandomForestClf.pickle', 'rb') # Open data file and read bytes.

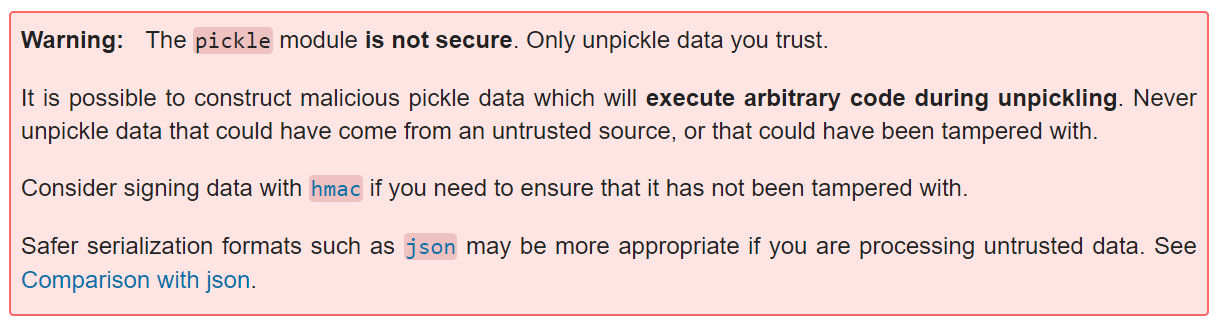
model = pickle.load(load\_clf) # To load the result from a file (Unpickle). load\_clf.close()

**Question:** What if there's more than one pickled object in the data file?

**Answer:** The load( ) method is only going to read those objects one at a time. So, the first call to load will get the first object, the next call will get the second object, and so on.

**Warning Message:**

Must trust the source of the pickle object for security reasons.



**Resource:**

Python Tutorial Pickle | http://docs.python.org/py3k/library/pickle.html

This page from the Python library reference describes the process of pickling and has links to another description of how to use pickle in Python.