mltooler

Release v0.001

Jan 21, 2020

Contents

We've all been there (haven't we?...) Spending hours trying to improve the accuracy of a model, switching out different scalers, transforming the data every which way. All in a bid to improve our score (and probably move up the kaggle ladder)

Mltools helps takes (some**) of the headache out of the process by automatically and iteratively improving the cross validation score of your chosen machine learning model.

Why not put your feet up and squeeze that last few percent out of what you thought might not even be possible.

CHAPTER 1

What has it got, and what does it do?

Contained within Mltooler is a selection of tools to help the user improve their model with little to no effort. If you want to squeeze out every last little bit you can then you have come to the right place.

SelfImprovingEstimator The star of the show - a pipeline wrapper for a selection of estimators and data preprocessing steps - Do you want to improve your cross validation score? Then this is the place to start.

VotingClassifier A voting classifier with added bang - calculate the best breakdown of estimator weights with ease.

StackingEnsemble A stacking estimator with a little magic.

HyperSearch A wrapper for hyoperopt - making parameter searing easier than ever. With timed early stopping and preset parameter grids.

RandomSearch A random parameter grid search.

EnembleEstimator The home for a stack of SelfImprovingEstimators - includes a **DIRTY** way to speed up the processing of multiple estimators tuning.

Please note this will eventually use threads and not the current method of spawning the code in seperate python consoles.

Warning: Usage Notes:

- Don't forget that "Garbage in equals garbage out" - mltooler is what it says on the tin, a tool. It's not magic and it cannot perform miracles with your data. If you are interested in performing miracles please consult your nearest bible.
- Due to the amount of possible pipeline steps, training and improvement can take a substantial amount of time, and probably best left running overnight for least amount of hair pulling or nail biting. Increased amount of input features or sheer volume of data decreases run speed significantly.
- Mltooler tries to recreate sklearn estimators in essence, they are not one in the same and some functions usually associated with sklearn algorithms will not work or do not exist.
- All code as been written in my spare time, as a project while I learn to code python. Please report and errors or irregularities and I will try to fix when possible.

- And finally... Although all efforts have been made to make mltooler as robust as possible, responsibility cannot be accepted for inconsistent or incorrect results.

1.1 Contact

If you want to contact me about this project or anything please dont hesitate to do so.

Email: lewis.morris@gmail.com

Facebook:  LewisMorris

Github: https://github.com/lewis-morris/mltooler

1.1.1 Getting Started with Mltooler

Installation using pip

Crack open your command prompt and type

```
pip install mltooler
```

That was easy wasn't it?

Mltooler requires the latest version of multiple 3rd party packages, make sure you update to the required version if asked.

Huge thanks goes out to the creators and contributors of these!

Importing Mltooler

Importing mltooler is again easy as pie - no bells and whistles here.

```
from mltooler import mltools as mt
```

Improving your first model

The SelfImprovingEstimator is what drives Mltooler. All you need is some training data, and the default settings will work their magic. This example shows the basics with minimal lines of code.

(See full API documentation for advanced usage and examples)

```
from mltooler import mltools as mt
sie = mt.SelfImprovingEstimator('knn')
sie.self_improve(X,y)
```

Once running the SelfImprovingEstimator will iteratively improve its cross validation score through a series of tests adding data transformations to its pipeline.

Once fit you can use the SelfImprovingEstimator like any scikit-learn estimator.

```
sie.fit(X_train,y_train)
preds = sie.predict(X_test)
```

1.1.2 Estimator Types

Self Improving Estimator

Ensemble Classifier

Voting Classifier

1.1.3 Full API Listing