

Case studies on Scikit-learn

1. Technology and Platform used for development

- a. What coding languages are used? Do you think the same languages would be used if the project was started today? What languages would you use for the project if starting it today?
 - b. What build system is used (e.g. Bazel, CMake, Meson)? What build tools / environment are needed to build (e.g. does it require Visual Studio or just GCC or ?)
 - c. What frameworks / libraries are used in the project? At least one of these projects don't use any external libraries or explicit threading, yet is noted for being the fastest in its category--in that case, what intrinsic language techniques is it using to get this speed.
- a. Scikit-learn is a package written by python and also used on python. Some core algorithm are written in Cython. As a ML-package, comparing with tensorflow, Scikit-learn mainly focuses on help users process data by their own, such as selecting features, compressing dimensions, and converting formats.

Python is the most popular language used in machine learning field, and as python's exclusive package, if Scikit-learn is built today, it will also use python. Also, I'd like to use python too, because it has a lot of convenient third-party libraries of mathematical operations and Structured data manipulation, such as with NumPy, SciPy, Pandas.


b. Scikit-learn is optimized by different authors, and I didn't find any resources showing the building system of Scikit-learn.

c. Scikit-learn's algorithm library is built on top of SciPy (an open source Python-based scientific computing toolkit) - you must install SciPy before you can use SciKit-learn. And also need the library NumPy.

2. Testing: describe unit/integration/module tests and the test framework

- How are they ensuring the testing is meaningful? Do they have code coverage metrics for example?
 - What CI platform(s) are they using (e.g. Travis-CI, AppVeyor)?
 - What computing platform combinations are tested on their CI?
E.g. Windows 10, Cygwin, Linux, Mac, GCC, Clang
- a. They use codecov to test the code coverage metrics.


build failing


 codecov 92%

circleci

passing

Branch: master ▾ scikit-learn / .codecov.yml

 lesteve Turn off codecov comments (#10146)

4 contributors 

23 lines (20 sloc) | 672 Bytes

```
1  comment: false
2
3  coverage:
4    status:
5      project:
6        default:
7          # Commits pushed to master should not make the overall
8          # project coverage decrease by more than 1%:
9          target: auto
10         threshold: 1%
11  patch:
12    default:
13      # Be tolerant on slight code coverage diff on PRs to limit
14      # noisy red coverage status on github PRs.
15      # Note The coverage stats are still uploaded
16      # to codecov so that PR reviewers can see uncovered lines
17      # in the github diff if they install the codecov browser
18      # extension:
19      # https://github.com/codecov/browser-extension
20      target: auto
21      threshold: 1%
22
```




2. Testing: describe unit/integration/module tests and the test framework

- How are they ensuring the testing is meaningful? Do they have code coverage metrics for example?
 - What CI platform(s) are they using (e.g. Travis-CI, AppVeyor)?
 - What computing platform combinations are tested on their CI?
E.g. Windows 10, Cygwin, Linux, Mac, GCC, Clang
- a. They use codecov to test the code coverage metrics.

- b. They use Travis-CI to test the units.

Branch: master ▾ [scikit-learn](#) / [.travis.yml](#)

 **adrinjalali** remove non-cron travis jobs (#13341)

20 contributors                   

43 lines (36 sloc) | 1.21 KB

```
1 # make it explicit that we favor the new container-based travis workers
2 sudo: false
3
4 language: python
5
6 cache:
7   apt: true
8   directories:
9     - $HOME/.cache/pip
10    - $HOME/.ccache
11
12 dist: xenial
13
14 - -
```

- c. Linux environment is tested on Travis-CI.

```
matrix:
  include:
    # Linux environment to test scikit-learn against numpy and scipy master
    # installed from their CI wheels in a virtualenv with the Python
    # interpreter provided by travis.
    - python: 3.7
      env: DISTRIB="scipy-dev" CHECK_WARNINGS="true"
      if: type = cron OR commit_message =~ /\[scipy-dev\]/
```

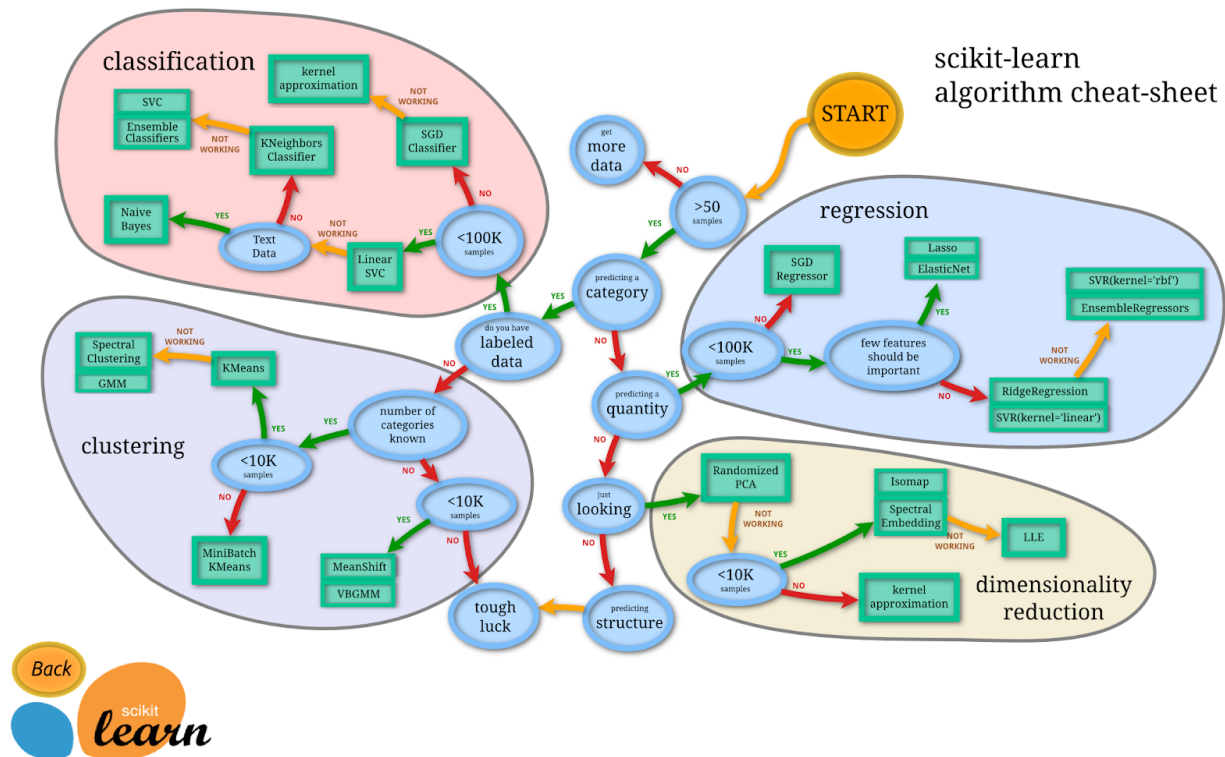
3. Software architecture in your own words, including:
- How would you add / edit functionality if you wanted to? How would one use this project from external projects, or is it only usable as a standalone program?
 - What parts of the software are asynchronous (if any)?
 - Please make diagrams as appropriate for your explanation

- d. How are separation of concerns and information hiding handled?
 - e. What architectural patterns are used
 - f. Does the project lean more towards object oriented or functional components
- a. Before wanting to add a new algorithm, which is usually a major and lengthy undertaking, we ought to start with “*Know issue*” on their github, in case some change may already have solutions. And they have some issue tags for the beginner to get familiar with contribution. Some “help wanted” issues also need contributors to solve.

Any sort of documentation is ok if we want to make some change, and we have to generate a HTML output by typing `make html` from the `doc/` directory. When you change the documentation in a pull request, CircleCI automatically builds it. Also, use `pytest` package to test the units.

Users can use Scikit-learn as a data process tool, which means it can be used as many single part or functions.

- b. Emm, we use it only when we call some specific functions.
- c. The diagram below is a classic diagram of scikit-learn structure of different function branches.



d. In the github, they make a great readme and seperate different part in different folder. And in the user guide, they give a lot of link from one question to another possible question you may come out, which is very clearly. (if this is the point of the problem d)

e. Actually I'm not sure if the Scikit-learn has a "architectural pattern", because "architectural pattern" is a concept used in software.

f. Yes, the Scikit-learn can be divided into six components: Classification, regression, clustering, data dimensionality reduction, model selection and data preprocessing. And every algorithm is packaged well in functions and the modules in scikit-learn are highly abstracted, increases the efficiency of the model, reducing the difficulty of batching and standardization (by using pipeline).

4. Analyze two defects in the project--e.g. open GitHub issue, support request tickets or feature request for the project
 - a. Does the issue require an architecture change, or is it just adding a new function or?
 - b. make a patch / pull request for the project to fix problem / add feature
 - a. It depends on the issue type, I think most issues don't have to change the architecture change.
 - b. It's hard to give a change or a optimization by yourself, because you have to communicate with others in the issue page on github first and may find someone to correct the bug or do some optimization. The team is fixed by the way. And one good side is you can train yourself from some easy issues proposed by others and correct them.
5. Making a demonstration application of the system, your own application showing how the software is used
- Using Scikit-learn to do some data preprocessing (standardization) and separate data, then do the logistic regression and create a classification report of prediction.

```
# -*-coding:utf-8-*-
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import ShuffleSplit
from sklearn.metrics import classification_report
# from sklearn.metrics import roc_auc_score

dataSet = load_iris()
data = dataSet['data'] # data
label = dataSet['target'] # data's label
feature = dataSet['feature_names'] # feature's name
target = dataSet['target_names'] # name of the label
print(target)
pd.set_option('display.max_columns', None)
```

```

df = pd.DataFrame(np.column_stack((data,label)).columns = np.append(feature,'label'))
print(df.head())# check the top 5 row

print(df.isnull().sum(axis=0).sort_values(ascending=False)/float(len(df)))# see the rate of missed value

# In sklearn's preprocessing, there is a function Imputer() to handle missing values.
# It provides median, mean, and mode to fill in missing values
# Fortunately, there are no missing values in our data set

print(df['label'].value_counts()) # check the rate of each label

StandardScaler().fit_transform(data) # z score standard

# use OvR doing multiple logistic regression
ss = ShuffleSplit(n_splits = 1, test_size = 0.2) # separate the dataset, 80% as training set
for tr,te in ss.split(data,label):
    xr = data[tr]
    xe = data[te]
    yr = label[tr]
    ye = label[te]
    clf = LogisticRegression(solver = 'lbfgs', multi_class = 'multinomial')
    clf.fit(xr,yr)
    predict = clf.predict(xe)
    print(classification_report(ye, predict))
# OvR regards multiple logistic regression as a binary logistic regression.
# The specific method is to select one class as a positive example each time,
# and the other categories as a negative case, and then do binary logistic regression
# to obtain the classification model of the class. Finally, multiple binary regression models are derived.
# The classification results are obtained according to the scores of each category.

```

The result is :

	precision	recall	f1-score	support
0	1.00	1.00	1.00	8
1	1.00	0.88	0.93	16
2	0.75	1.00	0.86	6
micro avg	0.93	0.93	0.93	30
macro avg	0.92	0.96	0.93	30
weighted avg	0.95	0.93	0.94	30
Process finished with exit code 0				

Reference:

Scikit-learn user guide:

https://scikit-learn.org/0.20/_downloads/scikit-learn-docs.pdf

An introduction to machine learning with scikit-learn:

<https://scikit-learn.org/stable/tutorial/basic/tutorial.html>

Introduction of Scikit-learn:

<https://en.wikipedia.org/wiki/Scikit-learn>