case-study-mdche001 Scikit-learn

case-study-mdche001 created by GitHub Classroom

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?

1).
scikit-learn is a Python module for machine learning built on top of SciPy and distributed under the 3-Clause BSD license. Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

2).
Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming. Many other paradigms are supported via extensions, including design by contract and logic programming.

For these features, I think it still suitable for today's requirements on scikit-learn development.

3).
If I need to focus on this project, I will prefer to use python as well.

First of all, python is my favorite programming language and it has some advantages as follows:

Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications.

Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management.

A Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

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 ?)

On Unix-like systems, you can simply type make in the top-level folder to build in-place and launch all the tests. Have a look at the Makefile for additional utilities.

On Mac OSX, the default C compiler, Apple-clang, on Mac OSX does not directly support OpenMP. The first solution to build scikit-learn is to install another C compiler such as gcc or llvm-clang. Another solution is to enable OpenMP support on the Apple-clang. (instruction link)

On Windows, to build scikit-learn on Windows you need a working C/C++ compiler in addition to numpy, scipy and setuptools. The building command depends on the architecture of the Python interpreter, 32-bit or 64-bit.(command instructions)

For build the scikit-learn on your computer, you will need Build Tools for Visual Studio 2017. Another alternative option is to use MinGW (a port of GCC to Windows OS) as an alternative to MSVC for 32-bit Python. Not that extensions built with mingw32 can be redistributed as reusable packages as they depend on GCC runtime libraries typically not installed on end-users environment.

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.

Scikit-learn requires:

- Python (>= 3.5),
- NumPy (>= 1.11),
- SciPy (>= 0.17).

Note: For installing on PyPy, PyPy3-v5.10+, Numpy 1.14.0+, and scipy 1.1.0+ are required. For PyPy, only installation instructions with pip apply.

Building Scikit-learn also requires

- Cython >=0.28.5
- OpenMP

Running tests requires

- pytest >=3.3.0

Some tests also require pandas.

Describe unit/integration/module tests and the test framework

a.How are they ensuring the testing is meaningful? Do they have code coverage metrics for example?

For high-quality unit testing, they use the pytest package. The tests are functions appropriately named, located in tests subdirectories, that check the validity of the algorithms and the different options of the code.

After the installing required packages, the scikit-learn packages can be tested by executing from outside the source directory. (pytest instruction) Scikit-learn can also be tested without having the package installed.(test command)

The full scikit-learn tests can be run using 'make' in the root folder. Alternatively, running 'pytest' in a folder will run all the tests of the corresponding subpackages.And the expected code coverage of new features is at least around 90%.The coverage of scikit-learn is provided by the codecov. As it displayed, the coverage of the recent 6 months is linear and the mainly decrease of the coverage is in **/sklearn/externals/joblib/** which is reflected by the `COVERAGE SUNBURST` .

b.c.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

CI: Travis is used for testing on Linux platforms. **Appveyor** is used for testing on Windows platforms. **CircleCI** is used to build the docs for viewing, for linting with flake8, and for testing with PyPy on Linux.

Software architecture in your own word

a.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?

Scikit learn provides users with varieties of APIs to modify their model, here is what I did to edit my function.

Users can modify different training models by setting the parameters of the model, now we can take the neural-network model as an example. Scikit learn provides users with the classifier module named **MLPClassifier,** the default parameters of the classifier shows as follow:

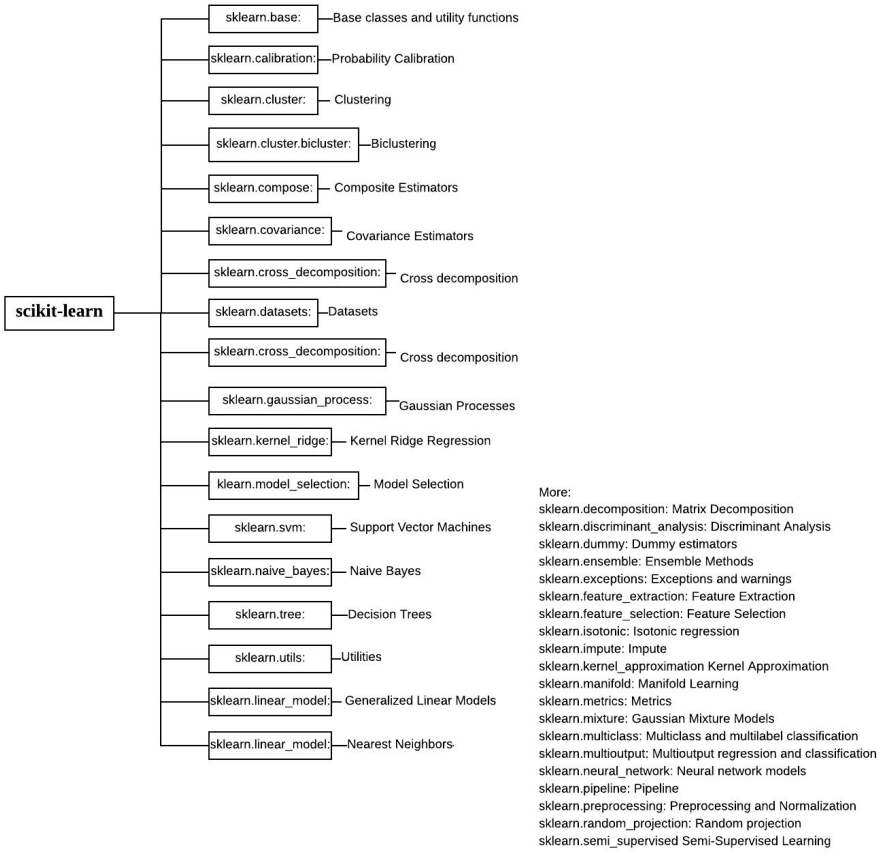
```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(13, 13, 13), learning_rate='constant',
       learning_rate_init=0.001, max_iter=500, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

You can modify the model as you prefer, like config some layers , set the batch size of the training or some stuff like that. All parameters' definition is involved in the manual.

These model can not only be the standalone program, but can be reloaded by built-in package named pickle or `joblib`.

For joblib: In the specific case of scikit-learn, it may be more interesting to use joblib's replacement for pickle (`joblib.dump` & `joblib.load`), which is more efficient on big data but it can only pickle to the disk and not to a string.(joblib Usage)

c. The diagram of the scikit-learn:



d.How are separation of concerns and information hiding handled?

In scikit-learn, users trained their model with their own datasets or datasets from sklearn.dataset. The parameters of the hidden layers can be set before the training and all hidden layers are anonymous. After training, they will get a trained model with weights which can be used to predict in the future. Thus, users will easily obtain the prediction of the test and then calculate the score of the model.

f. Does the project lean more towards object oriented or functional components

Based on my experience in scikit learn, all model will be transformed as objects. As a result, I think the scikit learn should be objected-oriented components.

Analyze two defects in the project

Some of the issue is to search for help by users and developers, which means there is no need for architecture modification. The aim of the rest is to add new features and bug fix, the kind of issue may lead to the change of the architecture. For example, Issue #13482: The sponsor of the issue shared new idea that they can provide some matrices related with the performance of the regression model. If necessary, they can offer to the scikit team to improve the user's experience of scikit-learn. Issue #13481: The issue reminds the scikit team of the shortcoming of the discrete_features == 'auto' and it may failure in the new version of numpy.

Making a demonstration application of the system

For simply explain how to use this project in your work, I design house price prediction model to illustrate the basic function of the scikit-learn. The following steps shows how that works:

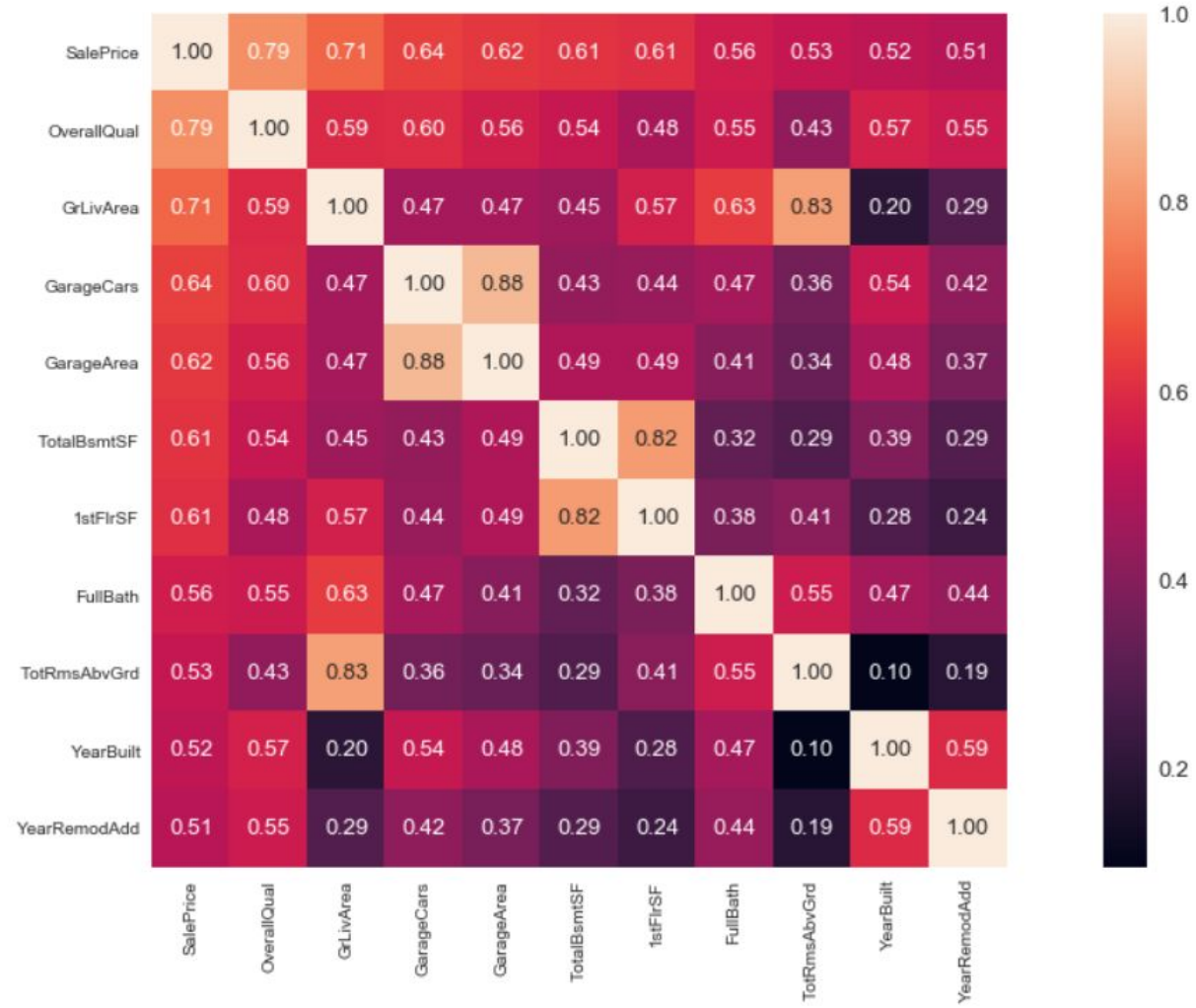
1). Loading dataset: In my part, the data is house.csv loaded by pandas library.

```
data_df = pd.read_csv("housedata.csv")
```

2). Analyzing the correlation between features and labels:

```
corrmat = data_df.corr().abs()
top_corr = corrmat[corrmat["SalePrice"]>0.5].sort_values(by = ["SalePrice"], ascending = False).index
cm = abs(np.corrcoef(data_df[top_corr].values.T))
f, ax = plt.subplots(figsize=(20, 9))
sns.set(font_scale=1.3)
hm = sns.heatmap(cm, cbar=True, annot=True,
                 square=True, fmt='.2f', annot_kws={'size': 13},
                 yticklabels=top_corr.values, xticklabels=top_corr.values);
data_df = data_df[top_corr]
```

Then using heatmap combined with correlation for objective analysis. That the correlation between variables and labels will easily appears by identifying different colors.



3). Split the data:

```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size = 0.2, random_state=10)
```

4). Model selection and calculate the score:

```
regressor = DecisionTreeRegressor(max_depth = depth)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
score = performance_metric(y_test, y_pred)
print("The R2 score is ", score)
```

```
The R2 score is  0.797192336397116
```

Another basic example for whole process is as follows:

The code is only an pseudo code session which means you have to fill in some essential elements to run it on your computers.

```
import pandas as pd
data = pd.read_csv("data.csv")

# Splitting the data into X and y
import numpy as np
X = np.array(data[['x1', 'x2']])
y = np.array(data['y'])

# Import statement for train_test_split
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

from sklearn.tree import DecisionTreeClassifier

# Define the model (with default hyperparameters)
clf = DecisionTreeClassifier(random_state=42)

# Fit the model
clf.fit(X_train, y_train)

# Make predictions using the unoptimized and model
train_predictions = clf.predict(X_train)
test_predictions = clf.predict(X_test)
```