
Regression2018 Documentation

Xu Xiao Cong

Jun 11, 2018

Contents

1	Research Logs	1
2	Indices and tables	7

1.1 Fri 9 Jun 2018

- Author(s): Zheng Le Wen
- Working Directory: *Research*

1.1.1 The Base Model Class

Today we wrote a base class `ModelBase` to ease subsequent model testing procedure. You can check it in the file *Research/model_test.py*.

`ModelBase` is basically a wrapper for all future classification models. It will load data for you, call your model's algorithms, and collect results. It perform the whole cross validation procedure for you. All you need to do to write your own model, is to:

- Derive it
- Override `setup` method. Define and initialize your model's parameters here.
- Override `train` method. This is where you will put the core learning algorithm of your model. Basically it is updating your params defined in `setup`. All needed knowledge is features and labels, and these are already given as method parameters.
- Override `predict` method. Use your trained model to predict and return labels here, with only features given.

A framework is as follow:

```
from model_test import ModelBase
import pandas as pd

class SimpleModel(ModelBase):
    def setup(self):
        # Define and initialize your model's parameters here
        pass
```

(continues on next page)

(continued from previous page)

```
def train(self, features, labels):
    # Specify how to train your model here
    # All data you need will be features and labels as given in parameters
    pass

def predict(self, features):
    # Compute and return your prediction here from given features
    # All data you need is features
    # Also model parameters should be already updated from train method

    # Initialize labels
    n = len(features)
    # Must return pandas Series
    labels = pd.Series([0]*n)
    return labels
```

For a running simple example, you can check the SimpleModel class in *Research/model_test.py*.

Then in main function, we can perform our experiment easily as follow:

```
from model_test import summary
# Create model
model = SimpleModel()
# Run model
results = model.run()
# Display results
summary(results)
```

Here summary method is used to format output the cross validation results.

You can simply run the script *Research/model_test.py* in a terminal to see all things running. Just run:

```
python model_test.py
```

under *Research/* directory.

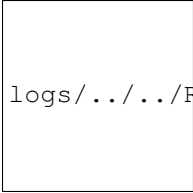
1.2 Sun 10 Jun 2018

- Working Directory: *Research*
- Updated Script: *script01.py*, *script02.py*

1.2.1 Exploration

Note: Exploration is in *script01.py*. Just run it in your shell.

For each gender, each pclass, we plot the survival ratio for people of different age range, namely 0~10, 10~20, 20~30, 30~40, 40~50, 50~60, 60~70, 70~80. The plot is shown below:



logs/../../Research/together.png

The first row shows survival distribution across different age range of males from different pclass. The second row shows woman's.

As can be seen, woman from first and middle class have high survive ratio that near 1.

We may use this survive distribution information to guess other people's destiny.

1.2.2 Experiment

Note: Experiment is in *script02.py*. Just run it under your shell.

In training stage, we compute the survival distribution of different sexes, different pclasses, across different age range, as mentioned in Exploration step. And then use this ratio as probability, we guess others' destinies in predicting stage.

The cross validation result is:

- correct rate: 73%

May be a little low.

1.2.3 Future Work

We need to try more general model with great predictive power.

1.3 Mon 11 Jun 2018

We want to incorporate cabin information into our model. But simply linearly separating the number in cabin code like "A12" into ranges would work well probably, since "A12" and "B12" may not align vertically. Besides, all cabins on A deck are located in one side of the ship, while cabins of other deck distributed across the ship.

So maybe we will need to hard coded the location information into the model.

I am gonna test against SVC, NuSVC and LinearSVC, BernoulliNB, GaussianNB models.

We are studying this [kernel](#) on kaggle which achieved over 82% on test set. It do not consider *Cabin* variable in its model because there are too many null values (over 77%). So even after we incorporate the *Cabin* information, at most only about 22% of the predictions will be improved. But we could try incorporating it, maybe.

The data preprocessing, exploration, and visualization and other many steps of the above kernel may be transferred to our report.

1.3.1 About Fusing models

If we want to combine individual models to create a *super* one, we should first plot the **correlation heatmap** with their predictions. By view their correlation heatmap, we can discover which of them are less correlated. Only little or even not correlated models could form stronger models when they are combined.

1.3.2 Today's Codings

Working Directories: *Research/*, *DataProcessor/*

Data Preprocessing

Under *DataProcessor/* folder, wrote 2 scripts:

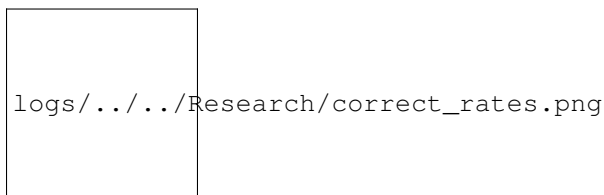
- *preprocessor.py*: Use `preprocess2` method to preprocess *train.csv* and *test.csv* data file. We filled in the missing data. We added `FamilySize`, `IsAlone`, `Title`, `FareBin`, `AgeBin` these new features, as mentioned in chapter 5 of [this notebook](#). And output the processed data into *Data/* directory, in the name of *preprocessed_train.csv* and *preprocessed_test.csv*.
- *process2.py*: Use script *basic_process.py* to generate cross validation set. Output directory is *Data/CrossValidation2/*

We will perform cross validation on this preprocessed and feature engineered data set later.

Model Selecting

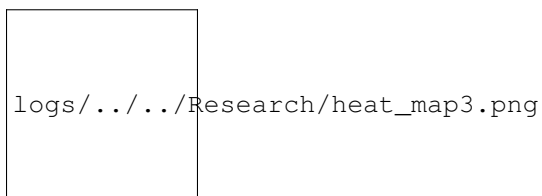
We only tested `SVC`, `NuSVC`, `LinearSVC`, `BernoulliNB`, `GaussianNB` against cross validation set, and the whole train-test suite. The scripts is in *Research/script04.py*.

The cross validation results is as follow (we only show correct rate):



As can be seen, `SVC` and `NuSVC` models achieve the best cross validation scores of 82%. While the naive bayes models achieve roughly 78%~79% scores.

Then we gather the predictions of these models, together with XiaoCong's neural network model's, YiHao's GLM model's, on main test data. And plot their correlation matrix as heatmap, as shown below:



as can be seen, `SVC` and `NuSVC` is highly correlated, while naive bayes and neural network is relatively less correlated with other models. This may indicate that they can be incorporate into our ultimate fused model. And we may only need one of `SVC` and `NuSVC` models, to save computing resources.

Note: the model correlation visualization procedure is coded in *DataProcessor/correlation.py*. It load in prediction file in the form of *modelname_predict.csv* under *Data/* folder, and then compute and display their correlation matrix (with only the lower triangle matrix is shown, as correlation matrix is symmetric).

1.3.3 On ModelBase Class

We modified `ModelBase` class so that it can perform both cross validation and main predicting procedures. We renamed `run` method to `run_cv` to indicate this is for cross validation. And created `run_main` method to run main predicting.

You can specify output file name when using `run_main`. The prediction is output into *Data/* directory.

And we made `summary` method return cross validation statistics.

All modification is coded in *Research/model_test2.py*.

This is for final project in the course *Multi-Variables Statistical Analysis* () by Prof. Huang (), in SYSU ().

Team mates:

- Xu Xiao Cong ()
- Zheng Le Wen ()
- Qin An Qi ()
- Yin Qian ()
- Ding Yi Hao ()

We are all undergraduate students in SYSU, major in applied math and statistics.

According to instructions from Prof. Huang, we decided that fulfilling the task in this [competition](#) on Kaggle as our project goal. You can check for further details in the link.

In this project, we are given information about passengers on the Titanic ship, including sex, age, class etc. For training set, we are told which passengers survived the catastrophe in 1912 when the ship sank in the North Atlantic Ocean after colliding with an iceberg. Then, we need to compute the survival probability of somebody in the test set with knowledge of his/her infos, and *predict* (or guess) if s/he lived eventually.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`