INFO370 Problem Set 6: classification

February 27, 2020

Introduction

This problem set is about classification, prediction, and training-testing split. It has the following aims:

- make you more experienced with logistic regression
- give you more experience with prediction
- teach you what is accuracy, precision, recall by asking you to compute those values manually
- make you slightly more confident with validation and testing-training-split.

The toughest ones can also try to understand and implement ROC curve.

Please submit a) your code (notebooks, rmd, whatever) *and* b) the results in a final output form (html or pdf).

Working together is fun and useful but you have to submit your own work. Discussing the solutions and problems with your classmates is all right but do not copy-paste their solution! Please list all your collaborators below:

1.
2. ...

Wisconsin Breast Cancer Dataset

You will work with Wisconsin Breast Cancer Dataset (WBCD), available at UCI Machine Learning Repository. You can download it from internet but rather use the files *wdbc.csv.bz2* from canvas (under *files/data*) where I have added the variable names. Check also out the brief description *wdbc_doc.txt*.

The data includes diagnosis of the tumor with "M" meaning cancer (*malignant*) and "B" no cancer (*benign*), and 10 features, describing physical properties of cell nuclei from biopsy samples. Each feature is represented three times, once for mean, once for standard deviation, and once for the worst values. Your task is to predict diagnosis based on this data using logistic regression.

1 Explore the data

As the first step, explore the data.

1. Load the data. You may drop *id* or just ignore it in the rest of your analysis. You may also convert *diagnosis* into a numerical variable.

2. Create a summary table where you show means, ranges, and number of missings for each variable. In addition, add (Pearson) correlation between the diagnosis and the corresponding feature. You may include more statistics you consider useful. Order this information by the absolute value of correlation, so the table might look like

variable	correlation	mean	min	max	std	missings
diagnosis	1.000	0.37	0	1	0.48	0
perimeter.worst	0.782	107.2	...			
...						

Here is an example code that achieves some ot these tasks but you can use a different approach.

```
import pandas as pd
wdbc = pd.read_csv("../data/wdbc.csv.bz2").drop('id', axis=1)
wdbc.diagnosis = (wdbc.diagnosis == "M").astype(int)
print("correlation with 'radius.mean':\n", wdbc.corrwith(wdbc['diagnosis']))
print("missings and such:\n",
      wdbc.apply(lambda col: pd.Series({'mean': col.mean(), 'std': col.std(),
                                        'missings': col.isnull().sum() })).T
      )
```

2 Which Model is Best?

Here we analyze how well do numerous logistic regression models. You will try including and excluding different variables and analyze accuracy and F-score. Your tasks are the following: split the data into training/validation chunks; fit the model on the training data; predict outcomes on training data, display; display the confusion matrix; and compute accuracy and F-score. Thereafter repeat the prediction, confusion matrix, and A/F on testing data. However, Accuracy and F-score you have to compute manually, without relying on the existing libraries.

Before you start, ensure you understand the confusion matrix and related measures *accuracy* A, *precision* P, *recall* R, and F-*score*. A good source is James et al. (2015, p 148), there is also some information in lecture notes, section 4.1.1.

1. let's start with a logistic regression model where you explain diagnosis with *concpoints.mean* and *fracdim.mean*. Extract these data and split the result into training and validation chunks.

   Here are a few lines of example code that do something along these lines:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
#
X = wdbc[['perimeter.worst']].values
#
Xtrain, Xvalid, ytrain, yvalid = train_test_split(X, y, test_size=0.2)
m = LogisticRegression(solver='lbfgs', C=1e9).fit(Xtrain, ytrain)
```

2. Now fit the model on *training data*

3. Predict the outcome on training data, and create a confusion matrix of the result.

   Hint: sklearn has a function `confusion_matrix` for computing confusion matrix. However, as confusion matrix is just a simple cross table, you can use `pd.crosstab` as well.

4. Based on the confusion matrix, calculate accuracy and F-score.

   Here I ask you to **implement A and F without using any other libraries**. Use your predictions, confusion matrix, and basic python functionality. Compute these values yourself!

5. As we explained in class, it is not a good idea to compute the goodness of model on training data. So now predict the outcome on validation data, and create a confusion matrix of the result.

6. Based on the confusion matrix, calculate accuracy and F-score.

7. Comment the results: did the model perform better on training or testing data? Does it hint about overfitting?

So this was about a single model. But what about all other possible models? There are many variables in data, you can add more of them (all of them?) into the model.

8. Repeat the above with a few other models: select different features or more features (you may select all of them too) and put into the model. Predict, display the confusion matrix and compute A and F on both trainig and validation dataset. (Use your own code, not libraries!)

9. Comment the results. Which model performed best in terms of A, F? Where the results on training data systematically better than on testing data?

3 Extra Credit: Implement ROC Curve (10 EC points)

Consult James et al. (2015, p 148) for what is the ROC curve, there is also some explanation about how to create it in lecture notes, section 4.1.1 (see section ROC curve). You will pick 2-3 models you estimated above and compare those using ROC curve.

1. split your data into training-validation parts (say, 80-20).

2. select 2-3 models you used above.

3. Train each model on training data.

4. Now instead of predicting the oucome category, predict *the probability that the case belongs to class "M"*. This can be done using `m.predict_proba` instead of `m.predict` if you use `sklearn`.

5. Now pick a sequence of probability thresholds between 0 and 1 (for instance, 0, 0.1, 0.2, ...). For each model and each threshold, treat your cases as "Malignant" if the predicted probability is at least as big as the threshold.

6. Based on these predictions, compute false positive rate and true positive rate for each threshold value. Plot these rates for each threshold and each model. This is your ROC curve.

7. Comment your results. Which model is the best based on ROC curves?

References

James, G., Witten, D., Hastie, T., Tibshirani, R., 2015. An Introduction to Statistical Learning with Applications in R. Springer.