CS3920/CS5920 Assessed Coursework Assignment 2

November 7, 2022

This assignment must be submitted by 21 November 2022, 16:00. Feedback will be provided within fifteen working days of the submission deadline.

Learning outcomes assessed

Be able to use and implement machine-learning algorithms, with the Lasso and inductive conformal prediction algorithms as examples. Have an understanding of ways to apply the ideas and algorithms of machine learning in industry and medicine.

Instructions

The coursework assignment must be completed strictly individually. You should not use in your submission any downloaded code or existing implementations of the learning algorithms. The submission is entirely electronic. You should submit one file (a Jupyter notebook) via the course's Moodle page.

Give your Jupyter notebook a reasonable name, such as `Lasso.ipynb`. It should contain all your code and your report containing the numerical results and, optionally, a discussion.

The file that you submit cannot be overwritten by anyone else, and it cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submissions after the deadline will be accepted but they will be automatically recorded as late and subject to College Regulations on late submissions. Please note that all your submissions will be graded anonymously; your name should not appear anywhere in your submission.

The deadline for submission is **Monday, 21 November, 16:00**. There can be no extension.

Note: All the work you submit should be solely your own work. Coursework submissions are routinely checked for this.

Coursework

This assignment requires an implementation of an inductive conformal predictor for regression and the ability to use the `Lasso` method, data preprocessing, and tools for parameter selection in `scikit-learn`. The method should be implemented in Python and submitted as a Jupyter notebook. Please leave all output produced by the system (i.e., do not remove the contents of cells like "Out [1]:").

Dataset

This assignment uses the dataset `diabetes` first used in the paper

> Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani (2004), Least Angle Regression (with discussion), *Annals of Statistics* **32**, 407–499.

For the description of the version of this dataset available in `scikit-learn`, run

```
print(diabetes.DESCR)
```

(as usual), where `diabetes` is my name for the dataset (you may use different names, but please make sure they are easy to remember). The `scikit-learn` function for loading this version of the dataset is `load_diabetes`.

For the main part of this assignment, you should use the original version of the dataset, which is given here (the file is called "Tab-delimited diabetes data (text file)") and which requires preprocessing. (Alternatively, you can download it from the module's Moodle page.)

You should perform the following steps:

1. Load the `scikit-learn` version of the `diabetes` dataset into your Jupyter notebook using the `load_diabetes` function.
2. Split the dataset into the training and test sets. You may use the function `train_test_split` in `scikit-learn`. Here and below use your birthday (in the format DDMM omitting leading zeros if any) as `random_state`.
3. What is the training and test R^2 for the `Lasso` model using the default parameters? How many features does this model use? What are the names of those features? Write the answers in your Jupyter notebook. Here and below, you are allowed to use any `scikit-learn` functions. (You may also use `numpy` and `matplotlib` functions.)
4. Now load the original `diabetes` dataset from the web page given above. Download the file

   `Tab-delimited diabetes data (text file)`

   by right-clicking on it. All the remaining tasks should be performed using this file (`diabetes.data`), which is the original `diabetes` dataset. The labels are given in the last column of the file `diabetes.data`.
5. Split the dataset into the training and test sets. Use your birthday (in the format DDMM) as `random_state`.
6. Repeat item 3 for the current dataset. Comment on the differences from what you saw in item 3.
7. Preprocess the training and test sets in the same way and avoiding data snooping. Use `StandardScaler`.

8. Repeat item 3 for the current training and test sets (which you should use in items 8–10). Are your current results closer to those in item 3 or item 6? Notice that *a priori* you would expect your current results to be closer to those in item 3, since the reason for different results in items 3 and 6 was that the former were for normalized data while the latter were for the original data. Is this expectation confirmed? If not, why?

9. Varying the regularization parameter α in the Lasso, plot the test R^2 vs the number of features used (i.e., those with non-zero coefficients). Try to make your plot as pretty as possible. (Obviously, it's subjective.) Which point on the curve do you prefer? (There is no unique correct answer to this question.) Give a brief explanation of your preference.

10. Choose the regularization parameter for the Lasso using cross-validation on the training set. Train the Lasso on the whole training set using the chosen values of the parameters. Report the resulting training and test R^2 and the number of features used. (As before, you are allowed to use any `scikit-learn` functions.)

11. Implement an inductive conformal predictor as follows:

    (a) Split the training set that you obtained in item 5 into two parts: the calibration set of size 99 and the rest of the training set (the training set proper). Use your birthday (in the format DDMM) as `random_state`.

    (b) Preprocess the training set proper, calibration set, and test set in the same way using `StandardScaler`. Namely, fit the scaler to the training set proper and then use it to transform all three.

    (c) Using the nonconformity measure $\alpha = |y - \hat{y}|$, where y is the true label and $\hat{y}$ is its prediction given the training set proper, for each test sample compute the prediction interval for it. Do this for significance levels 5% and 20%. For each of these significance levels compute:

    - the length of the prediction intervals for the test samples
    - and the test error rate of your inductive conformal predictor[1].

    For computing the predictions $\hat{y}$, use the Lasso with parameters chosen by cross-validation on the training set proper.

All results should be given in your Jupyter notebook (see above for a description of what to include). Make sure to include the following numbers:

(a) The training and test R^2 for the `Lasso` model with default parameters on the `scikit-learn` version of `diabetes` and the number of features used.

[1]An inductive conformal predictor makes an error if its prediction interval fails to cover the true label. The test error rate is the number of errors made on the test samples divided by the size of the test set.

(b) The training and test R^2 for the `Lasso` model with default parameters on the original version of `diabetes` and the number of features used.

(c) The training and test R^2 for the `Lasso` model with default parameters on your version of `diabetes` (i.e., on the original version of `diabetes` with the features normalized by you); the number of features used.

(d) The training and test R^2 for the `Lasso` model with the best parameters chosen by cross-validation on your version of `diabetes`; the number of features used.

(e) The lengths of prediction intervals and their test error rates at significance levels 5% and 20% (if you are implementing an inductive conformal predictor).

There should be 16 numbers overall, plus your explanations (including the names of chosen features) and a plot.

Inductive conformal prediction is described in Chapter 6. You are not allowed to use any existing implementations of inductive conformal prediction, but you are allowed to use the functions available in `scikit-learn`, such as `Lasso`, and `NumPy`, such as `sort`.

Marking criteria

To be awarded full marks you need both to submit correct code and to obtain correct results on the given dataset. Even if your results are not correct, marks will be awarded for correct or partially correct code. An ideal implementation of items 1–10 will give you 70%. The remaining 30% will be awarded for implementing an inductive conformal predictor (item 11).

Extra marks

There are several ways to get extra marks (at most 10%) that will be added to your overall mark (the sum will be truncated to 100% if necessary). Extra marks will be given for:

- Interesting observations about Lasso.
- Careful breaking of ties among the nonconformity scores of the calibration samples (remember that so far our assumption has been that they are all different, in the case of regression).
- Other ways of testing the validity of the inductive conformal predictor.

Remember that you should submit only one file, your Jupyter notebook.

Appendix Further details for task 11(c)

You can follow the solution to the exercise on slide 51 of Chapter 6, except that Nearest Neighbour needs to be replaced by the Lasso as the underlying algorithm used for producing the predictions $\hat{y}$. These are the steps for the inductive conformal predictor in task 11(c):

1. Fit the Lasso to the training set proper.
2. Compute the α_i for the calibration set by the formula $\alpha_i = |y_i - \hat{y}_i|$ (y_i being the true label and $\hat{y}_i$ being the Lasso prediction).
3. Compute k and c, as explained on slides 47–48 of Chapter 6.
4. Finally, compute the prediction set for each test sample as $[\hat{y} - c, \hat{y} + c]$, $\hat{y}$ being the Lasso prediction for this test sample.