CPSC 483 - Introduction to Machine Learning

Project 3, Fall 2020

due October 26 (Section 02) / October 29 (Section 01)

Last updated Saturday October 10, 10:30 pm PDT

Having looked "under the hood" at NumPy implementations of linear and polynomial regression in Project 2, we return to scikit-learn to see what it can offer in terms of automating cross-validation procedures.

The project may be completed individually or in a group of no more than three (3) people. All students on the team must be enrolled in the same section of the course.

Platforms

The platform requirements for this project are the same as for Project 1 and Project 2.

Libraries

You will need scikit-learn to obtain the data, build models, and run cross-validation. You may also wish to use pandas DataFrames to examine and work with the data, but this is not a requirement.

You may reuse code from the Jupyter notebooks accompanying the textbook and from the documentation for the libraries. All other code and the results of experiments should be your own.

Dataset

The scikit-learn `sklearn.datasets` module includes some small datasets for experimentation. In this project we will turn the tables on the Boston house prices dataset. The original use of the dataset was to try and predict the median value of a home given several features of its neighborhood. One of features, however, is CRIM, the per-capita crime rate by town.

In this project we will engage in some amateur predictive policing, attempting to predict the crime rate using the other features in the dataset.

See the section on scikit-learn in Sergiy Kolesnikov's blog article Datasets in Python to see how to load this dataset and examine it using pandas DataFrames.

Experiments

Run the following experiments in a Jupyter notebook, performing each action in a [code cell]() and answering each question in a [Markdown cell]().

1. Load and examine the Boston dataset's features, target values, and description.

2. Save CRIM as the new target value ***t***, and drop the column CRIM from ***X***. Add the target value MEDV to ***X***.

3. Use `sklearn.model_selection.train_test_split()` to split the features and target values into separate training and test sets. Use 80% of the original data as a training set, and 20% for testing.

4. Create and `fit()` an `sklearn.linear_model.LinearRegression` to the training set.

5. Use the `predict()` method of the model to find the response for each value in the test set, and `sklearn.metrics.mean_squared_error()`, to find the training and test MSE.

6. By itself, the MSE doesn't tell us much. Use the `score()` method of the model to find the R^2 values for the training and test data.

   R^2, the *coefficient of determination*, measures the proportion of variability in the target ***t*** that can be explained using the features in ***X***. A value near 1 indicates that most of the variability in the response has been explained by the regression, while a value near 0 indicates that the regression does not explain much of the variability. See Section 3.1.3 of *An Introduction to Statistical Learning* for details.

   Given the R^2 scores, how well did our model do?

7. Let's see if we can fit the data better with a more flexible model. Scikit-learn can construct polynomial features for us using sklearn.preprocessing.PolynomialFeatures (though note that this includes interaction features as well; you saw in Project 2 that purely polynomial features can easily be constructed using `numpy.hstack()`).

   Add degree-2 polynomial features, then fit a new linear model. Compare the training and test MSE and R^2 scores. Do we seem to be overfitting?

8. Regularization would allow us to construct a model of intermediate complexity by penalizing large values for the coefficients. Scikit-learn provides this as `sklearn.linear_model.Ridge`. The parameter `alpha` corresponds to λ as shown in the textbook. For now, leave it set to the default value of 1.0, and fit the model to the degree-2 polynomial features. Don't forget to normalize your features.

Once again, compare the training and test MSE and R^2 scores. Is this model an improvement?

9. We used the default penalty value of 1.0 in the previous experiment, but there's no reason to believe that this is optimal. Use `sklearn.linear_model.RidgeCV` to find an optimal value for `alpha`. How does this compare to experiment *(8)*?

Submission

Submit your Jupyter `.ipynb` notebook file through Canvas before class on the due date. Your notebook should include the usual identifying information found in a `README.TXT` file.

If the assignment is completed by a team, only one submission is required. Be certain to identify the names of all students on your team at the top of the notebook.