Satej Soman
CAPP30254: Machine Learning for Public Policy
Spring 2019

HW 5
MACHINE LEARNING PIPELINE
ADDITIONAL IMPROVEMENTS

Contents

Notes

- Representative code snippets are interspersed with analysis and explanations below; all code is available on GitHub: `https://github.com/satejsoman/capp30254/tree/master/hw5`.
- The pipeline library lives in the `code/pipeline` directory, while the sample application which imports the library is `code/donors_choose.py`.
- The generated table of hyperparameters is uploaded as a separate CSV; it also is available on GitHub as `code/output/donors_choose-LATEST/evaluations.csv`.

1 Pipeline Improvements

The following improvements have been made to the `pipeline` library:

1.1 Code Structure

- **hyperparameter grid**: A concept of a hyperparameter grid is formalized in the `pipeline.Grid` class. For each model, the parameters are varied, and the `Grid` class ensures the Cartesian product of all parameter values is trained and tested.

  Example configuration:

```
models:
  LogisticRegression:
    C: [0.01, 0.1, 1, 10, 100]
    penalty: ["l1", "l2"]
    n_jobs: [-1]

  KNeighborsClassifier:
    n_neighbors: [10, 50, 100]
    n_jobs: [-1]

  DecisionTreeClassifier:
    max_depth: [null, 1, 5, 10, 50, 100]
```

- **extended configuration**: Most pipeline settings are now read from a `config.yml` file instead of being hard-coded.
- **separate imputation for test and training sets**: Imputation of missing values is done per training and testing dataset instead of on the entire dataset.
- **metrics at *k***: Metrics are calculated by varying the top k% of scored entities rather than by thresholding. See `code/pipeline/evaluation.py` for implementation.
- **unit tests**: Unit tests are located at `code/pipeline/tests`. By running these tests (using either `pytest` or `python -m unittest discover`), we can confirm that:
  - the imputation of missing values runs only on each training and testing set
  - the splitting of datasets matches the specification in the pipeline configuration
  - a sample pipeline is working as a smoketest for any new changes made to the `pipeline` library

1.2 Methodology

- **training labels**: The label for the analyzed DataChoose data set has been updated so it a label of `1` indicates a project was *not* funded within `60` days.
- **test/train buffers**: There are `60` day gaps between the end of each training set and the beginning of its corresponding test set to account for the predicted outcome occurring after the end of the training period.

1.3 Models

- **SVM**: A linear support-vector machine is included in the models trained by the pipeline.
- **meaningful parameters**: Decision tree classifiers are now run varying maximum tree depth instead of information content measure, and the size of ensemble in random forests is also varied.

2 Analysis

Please see Homework 3 for initial analysis and data exploration. The following analysis focuses on new results as a result of pipeline changes. As a brief summary, Table 1 lists the features passed to the classifiers, and Table 2 lists the classifiers tested (including information on which parameters were controlled).

FEATURE	TYPE	DESCRIPTION
`school_city_categorical`	categorical	city in which school is located
`school_state_categorical`	categorical	state in which school is located
`primary_focus_subject_categorical`	categorical	broad subject on which the project focuses
`primary_focus_area_categorical`	categorical	specific area on which the project focuses
`resource_type_categorical`	categorical	type of resources asked for in project
`poverty_level_categorical`	categorical	socioeconomic indicator of school
`grade_level_categorical`	categorical	grade level of students affected by project
`school_charter_binary`	binary	whether the school is a charter school
`school_magnet_binary`	binary	whether the school is a magnet school
`students_reached_clean_scaled`	continuous	the number of students reached, scaled to $[0, 1]$
`total_price_including_optional_support_scaled`	continuous	the total price of the project, scaled to $[0, 1]$
`eligible_double_your_impact_match_binary`	binary	eligible for donation matching
`month_posted`	categorical	the month in which the project was posted
`funded_in_60_days`	binary	**column to predict**

Table 1: Features passed to machine learning models.

MODEL	PARAMETER	DESCRIPTION	VALUES
logistic regression	penalty	dimension of regularization function	L^1, L^2
	cost (C)	model penalization for misclassification	10^{-2}, 10^{-1}, 1, 10, 10^2
SVM	cost (C)	model penalization for misclassification	10^{-2}, 10^{-1}, 1, 10, 10^2
k-nearest neighbors	neighbors (k)	number of points to use for classification	10, 50, 100
decision tree	depth (d)	maximum depth of tree	1, 5, 10, 50, 100, unbounded
random forest	estimators (n)	ensemble size	10, 100, 1000
	depth (d)	maximum depth of tree	1, 5, 10, 50, 100, unbounded
gradient boosted classifier	learning rate (α)	gradient vector multiplier	0.1, 0.5, 2.0
bagging	sample fraction	bootstrap sample fraction	0.1, 0.5, 1.0

Table 2: Classifiers and parameters tested.

With the label properly calculated (i.e. `1` indicates failure to receive funding), we can answer the following questions:

- **Which classifier does better on which metrics?**

  Across both precision and recall for at the top 1%, 2% and 5%, the SVM and logistic regression classifier families outperform the other types of classifiers. This is not surprising, as the approach taken here resembles the causal inference worldview, where a small number of features is sufficient. Creating hundreds of features would allow the decision tree-based classifiers to perform much better.

- **How do results change over time?**

  Almost all classifiers show increasing performance over time; this is understandable, as models trained on the larger test sets should have higher performance since they are able to see more data points in the training phase.

- **What would be your recommendation to someone who's working on this model to identify 5% of posted projects that are at highest risk of not getting fully funded to intervene with, which model should they decide to go forward with and deploy?**

  I would recommend a support-vector machine with an *L2* penalty and further investigation to find optimal cost. While some classifier families perform strictly better when measured by precision at 5%, SVM classifiers perform the best when measured by precision at 2%. Presumably, the budget for intervention would be spent first on the projects at highest risk of failure. The higher precision of the SVM classifiers at 2% of the project population indicates ranking by the scores from this model would lead to the most effective interventions. Figure 2 shows the precision-recall curve for an SVM across all three test sets.

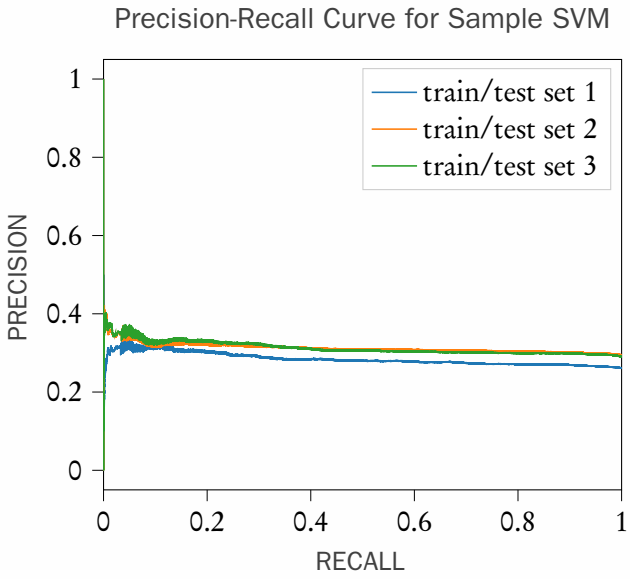

Figure 1: The precision-recall curve across the whole project population for a sample SVM classifier, evaluated on all three validation sets.