

**Master of Technology in**

Knowledge Engineering

**Computational Intelligence I**

Bharat Nagaraju - A0178258N

Vigneshram Selvaraj – A0178215A

Laxman Singh – A0178223E

Blog Feedback – Predictive Modeling (Regression)

Predict Number of Comments on a Blog.

**Objective:**

Train a group of different types of NNs using different NN tools to solve a regression problem using a dataset holding at least 5000+ records and 50+ variables. Build NN ensembles and compare the results.

1. **Background:**

The dataset originates from Budapest University of Technology and Economics’ research on blog posts. Data were collected by crawling raw HTML-documents. A base time was chosen and blog posts published at most 72 hours before that were selected and parsed. The task is to predict the number of comments for a posted blog after 24 hours of the baseline. In the train data, the base time were in year 2010 and 2011. In the test data, the base time were in February and March 2012. The dataset has been referred from [UCI Repository](https://archive.ics.uci.edu/ml/datasets/BlogFeedback)

1. **Data Exploration and Analysis**

In the Blog Feedback Data Set used here, there are 60 test files and one training file in the dataset. The first 280 columns in the dataset are independent variables, including number of comments/links in Different time periods in the last 72 hours, aggregation of such features by source, words contained in the blog posts, weekday information and parent page information.

The last column of training data is the dependent variable – number of comments in the next 24h. The train data has 52397 observations, and the test data has 7624 observations. The train data was generated from different base times that may temporally overlap. Therefore, if the data is simply split the train into disjoint partitions, the underlying time intervals may overlap. Therefore, temporally disjoint train and test splits should be used in order to ensure that the evaluation is fair.

1. **Preprocessing & Feature selection:**

Combine the 60 files to form the training data set. Apply Log transform on the output variable and drop the Continuous variables with no variation are dropped (Col numbers: 8, 13, 28, 33, 38, 40, 43, 50, and 278) on the training and the test data set.

This Training data set will be used to train different models and validated against the test dataset.

1. **Model Construction Approach:**

The flowchart of the model construction is:



1. **Model construction**

With new features as the input of the model, our next target is to build a model that could achieve the lowest value of the loss function on the test dataset. Thus, we experiment with different models that are popular for regression problem and then tune them to perform well on the test set. We compare their performance and select the best one or two models to construct the final ensemble model.

The models are fitted on the log-transformed responses. Model performances are measured by mean squared error. That is, for original response *y* and prediction ƒ̂(*x*), the error is calculated by (1 / *n*)∑[ln(1 + *y*)−ƒ̂(*x*)]².



1. **Basic Models**

We began by tuning some basic models on the dataset. Their performance are listed below:

|  |  |
| --- | --- |
| Model | Error |
| *k*-NN | **0.6407074** |
| LASSO | 0.6359344 |
| SVM | 0.4454636 |
| Random Forest | 0.3975424 |
| Boosting | 0.3840064 |

1. **Weighted Average**

Next average results from different models. Train 5 random forests and 5 boosting model. The following table listed the performances of the best of them, their averages, and the weighted average of 0.25 RF + 0.75 BST:

|  |  |
| --- | --- |
| Model | Error |
| Best RF | 0.3964153 |
| Best BST | 0.3826422 |
| Mean RF | 0.3968036 |
| Mean BST | 0.3829347 |
| Best RF + BST | 0.3805721 |
| Mean RF + BST | 0.3811796 |

It can be seen that the weighted average model outperforms any of the individual model.

1. **Stacked Generalization**

For the ensemble algorithm called stacked generalization, Pick 5 models for "level-0 generalizers": *k*-NN, LASSO, SVM, RF, and BST.

The procedure is then as follows:

1. Split the training dataset in to *k* folds.
2. For *ith* fold of training data, train models with each of the level-0 learners on the rest of the training data.
3. Make predictions using the models we trained:

* Make predictions on *ith* fold of data.
* Make predictions on the test set.

1. Repeat 2 & 3 until all *k* folds of training data are used on the models.
2. Generate the level-1 data using predictions obtained:

* Generate level-1 training data by concatenating the predictions made on each fold of the training data.
* Generate level-1 test data by averaging the predictions made on the test set.

1. Train a level-1 generalizer using the level-1 data and the correct responses, make predictions.

The level-1 generalizer used in this example is elastic-net regression (α=0.2). The following table listed the performances of the level-1 test data and the stacked model:

|  |  |
| --- | --- |
| Model | Error |
| LV-1 *k*-NN | 0.631110 |
| LV-1 LASSO | 0.624957 |
| LV-1 SVM | 0.443895 |
| LV-1 RF | 0.393018 |
| LV-1 BST | 0.379804 |
| Stacked | 0.383522 |

1. **Conclusion**

1) The individual networks are performing just ok

2) Averaging Ensembles provide the best results for this dataset compared to averaging of Random forests or Gradient Boosting or any other individual models

3) Individual Model – Gradient Boosting performs almost on par with Stacked / Averaged Ensemble but again it’s a form of ensemble.

1. **References:**

<https://machinelearningmastery.com/machine-learning-ensembles-with-r/>

<https://www.analyticsvidhya.com/blog/2017/02/introduction-to-ensembling-along-with-implementation-in-r/>

<https://mlwave.com/kaggle-ensembling-guide/>

<https://www.datacamp.com/community/tutorials/ensemble-r-machine-learning>