***Short Answer Scoring –***

***EXPLANATION OF “Gxav” SOLUTION***

Name: Xavier Conort

Email: [xavier.conort@gear-analytics.com](mailto:xavier.conort@gear-analytics.com)

Competition: ASAP – Phase2 - Short Answer Scoring

# Summary

The solution was designed for the competition ASAP – Phase2 - Short Answer Scoring where competitors had to develop a scoring algorithm for student-written short-answer responses.

The competition data contained 10 prompt sets. On average, each answer is approximately 50 words in length.  Some are more dependent upon source materials than others, and the answers cover a broad range of disciplines (from English Language Arts to Science).

Competitors were provided with:

* training data for each prompt.  Most training sets consisted of about 1,800 responses that have been randomly selected from a sample of approximately 3,000
* test data that contained approximately 6,000 new responses (600 per data set), randomly selected for blind evaluation

“Gxav” solution was developed on open source R and consists of 6 ensembles of 81 individual models, all trained per answer set, using:

* the following Machine Learning algorithms all available in R
  + Regularized Generalized Linear Models
    - Gaussian Error
    - 1-class classification
    - Multinomial Regression
  + Support Vector Machine (SVM) with linear and radial kernel
    - Regression
    - 1-class classification
    - and multi-class classification
  + Random Forest (RF)
  + Gradient Boosting Machine (GBM) with Gaussian error
* Score1 and score2 to train the algorithms
* 2 different flags (“good” and “bad” answers) to train 1-class classification algorithms
* 1 set of proxies
* 2 sets of document term matrix
  + one where stemming Is done on n-grams
  + one where stemming is done before producing n-grams and n-grams are produced on a selection of stemmed 1-grams
* 3 scaling techniques based on
  + the delta tf-idf metric
  + the bi-normal separation metric
  + ridit coding
* 2 reduction techniques
  + via regularized regression
  + via PCA

To blend different fits together, the holdout predictions of the 81 models were given as inputs to second-level learning algorithms (approach known as stacking).

* A 5 folds cross validation approach was adopted to produce holdout predictions for the stacking procedure
* 2 algorithms were used at the second-level
  + Regularized Generalized Linear Models with Gaussian error
  + Generalized Additive Models (GAM) with Gaussian error
* Ordinary Least Squares (LM) was used to blend the 2 ensembles at the third-level

The 3 ensembles were adjusted to produce the best SQWKappa using an optimization algorithm based on Nelder–Mead to obtain coefficients of a

* + 1d polynomial
  + 2d polynomial

As a final solution, we selected the best adjusted ensemble (among the 6 adjusted ensembles) using the 5 folds cv SQWKappa scores obtained for each adjusted ensemble per answer set.

Note the solution is fully automated and has not been customized to any answer set except for :

* the creation of dummies which label good and poor answers which were set as function of the distribution of scores for each sets
* set 1: answers with the last character not equal to "!","\"",".","?",")" and with less than 100 characters were treated separately. This was due to the presence of truncated answers for set1.
* set2 : the final solution was selected manually. This was done to ensure a distribution of the predictions for the validation set in line with the distribution of the holdout predictions obtained for the training set. The manual selection for set 2 improved slightly the public leaderboard score.

# Hardware and Software used

The solution was executed on a MacBook Pro using Mac OS X (version 10.6.8) 2.53 GHz Intel Core i5 - 4GB RAM.

R VERSION USED : 2.15.1 (2012-06-22) -- "Roasted Marshmallows"

R add-on packages:

* SOAR # to manage memory usage
* Hmisc and MASS # for descriptive analysis
* Matrix # to handle matrix
* RTextTools # to stem words
* tau # to produce n-grams
* tm # to remove duplicated blanks and punctuation marks
* clim.pact # to convert to lower case
* glmnet # to train Regularized generalized linear models
* glmnetcr # to train a Regularized multinomial generalized linear model
* e1071 # to train Support Vector Machine
* gbm # to train Gradient Boosting Machine
* randomForest # to train Random Forest
* mgcv # to train Generalized Additive Models

# Feature engineering

* We read in R the competition files specifying a file encoding “windows-1252”
* For answer set 1, we removed from the training set, answers with the last character not equal to "!","\"",".","?",")" and with less than 100 characters. This was motivated by the presence of truncated answers.

# Dummies which label good/poor answers

* We created dummies which label good and poor answers using both score1 and score2

|  |  |  |
| --- | --- | --- |
| Answer set | poor | good |
| Set1 | 0 | 3 |
| Set2 | 0 | 3 |
| Set3 | 0 | 2 |
| Set4 | 0 | 2 |
| Set5 | 0 | 1-3 |
| Set6 | 0 | 1-3 |
| Set7 | 0 | 2 |
| Set8 | 0 | 2 |
| Set8 | 0 | 2 |
| Set8 | 0 | 2 |

* 4 dummies were produced per answer set
  + isgood is the label for answers with good score1
  + isbad is the label for answers with poor score1
  + isgood2 is the label for answers with good score2
  + isbad2 is the label for answers with poor score2
* This was used
  + to train 1-class classification algorithms
  + to compute metrics to scale document-term matrix

# Proxies

* We built the following proxies

|  |  |
| --- | --- |
| **Proxy name** | **Description** |
| NCHAR | nb of characters |
| NW | nb of words |
| Q90 | size of words 90% quantile |
| Q95 | size of words 95% quantile |
| Q99 | size of words 99% quantile |
| PPrV | nb of precise verbs / nb of words |
| PTransW | nb of transition words / nb of words |
| PErr | nb of spelling errors / nb of words |
| QuotM | nb of quotes |
| BK | nb of brakets |
| CM | nb of commas / nb of words |
| POS | nb of " 's" |

* Some proxies were generated thanks to external data

# External data

To generate proxies, we used external and "homemade" data organized in 7 files.

We checked with Ben Hammer from Kaggle via the forum thread “External Data” of the previous competition whether those additional external data were permissible.

|  |  |  |
| --- | --- | --- |
| File | Source | Purpose |
| precise\_verbs.csv | http://www.owlnet.rice.edu/~cainp  roj/writingtips/preciseverbs.html | To estimate usage of precise verbs |
| transition\_words.csv | <http://www.smart-words.org> | To estimate usage of transition words |
| norvig.txt | <http://norvig.com/big.txt> (concatenation by Peter Norvig of several public domain books from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus) | To estimate spelling errors |
| gutenberg.txt | Ebooks from <http://gutenberg.org>  Creation Myths of Primitive America, by Jeremiah Curtin  The Legend of Sleepy Hollow, by Washington Irving  Tender Buttons, by Gertrude Stein  Three Soldiers, by John Dos Passos | To estimate spelling errors |
| essay\_inst.txt | Combination of the 10 sets instructions | To estimate spelling errors |
| academic words.csv | <http://www.uefap.com/vocab/select/awl.htm> | To estimate spelling errors |
| my\_list.txt | own list which was not updated for the competition. | To estimate spelling errors |

# Data transformation

* Some manipulation was necessary to:
  + produce n-grams
  + stem words and n-grams
  + remove duplicated blanks
  + convert all words into lower case
  + obtain punctuation structure consistent across all answer sets
  + remove punctuation
  + splitting the answers into words

# Document term matrix

* For each answer set, we produced 3 incidence document term matrix (binary matrix that correspond to presence/absence of a term in an answer)
  + One (bin\_mat.Ex) is based on stemmed 1-gram, 2-grams, 3-grams and 4-grams which appeared at least in 20 answers globally and in 5 answers at the answer set level
  + One (Bin\_mat.1.Ex) is based on stemmed words which appeared at least in 5 answers globally
  + One (bin\_mat.2.Ex) is based on n-grams of terms which were selected in a regularized regression trained on Bin\_mat.1.Ex. Rare n-grams (less than 5 times) were removed. We produced a different matrix for each round of the cross validation procedure (to avoid overfitting).
  + Note that only bin\_mat.Ex and bin\_mat.2.Ex are used to train futher algorithms. The second matrix was used only to produce the third document term matrix.

# Scaled matrix

* We used 3 different techniques :
  + delta tfidf
  + bi-Normal separation
  + Ridit coding
* The use of delta tf-idf and bi-normal separation metrics was inspired by the 2 following papers:
* Delta TFIDF: An Improved Feature Space for Sentiment Analysis. Justin Martineau, and Tim Finin. <http://ebiquity.umbc.edu/_file_directory_/papers/446.pdf>
* BNS Feature Scaling: An Improved Representation over TF·IDF for SVM Text Classification. George Forman HP Laboratories HPL-2007-32R1. <http://www.hpl.hp.com/techreports/2007/HPL-2007-32R1.pdf>
* To compute the metrics per answer set , we wrote our own function “METRICS”. When performing 5folds cross-validation, we repeated the metrics evaluation for each round of the cross-validation procedure.
* The metrics were computed using the good and bad dummies created previously
  + metrics.Ex using isgood dummy and the matrix bin\_mat.Ex
  + metrics.IB.Ex using isbad dummy and the matrix bin\_mat.Ex
  + metrics.2.Ex using isgood dummy and the matrix bin\_mat.2.Ex
  + metrics.IB.2.Ex using isbad dummy and the matrix bin\_mat.2.Ex
* The matrix were scaled using the previous metrics

Delta tf-idf

* + scal\_mat1.Ex using delta tf-idf in metrics.Ex and bin\_mat.Ex
  + scal.IB\_mat1.Ex using delta tf-idf in metrics.IB.Ex and bin\_mat.Ex
  + scal.mat1.2.Ex using delta tf-idf in metrics.2.Ex and bin\_mat.2.Ex
  + scal.IB\_mat1.2.Ex using delta tf-idf in metrics.IB.2.Ex and bin\_mat.2.Ex

Bi-normal separation (bns)

* + scal\_mat3.Ex using bns in metrics.Ex and bin\_mat.Ex
  + scal.IB\_mat3.Ex using bns in metrics.IB.Ex and bin\_mat.Ex
  + scal.mat3.2.Ex using bns in metrics.2.Ex and bin\_mat.2.Ex
  + scal.IB\_mat3.2.Ex using bns in metrics.IB.2.Ex and bin\_mat.2.Ex
* Ridit scoring is a way of recoding variables in a data set so that one has a measure not of their absolute values but their positions in the distribution of observed values.
* 3 matrix were produced
  + Ridit\_mat.Ex, ridit coding of bin\_mat. Ex
  + Ridit\_mat.1.Ex, ridit coding of Bin\_mat.1.Ex
  + Ridit\_mat.2.Ex, ridit coding of Bin\_mat.2.Ex

# Modelling techniques and training

The following Machine Learning algorithms were trained at the first level

* + Regularized Generalized Linear Models (Glmnet)
    - Gaussian Error
    - 1-class classification
    - Multinomial Regression
  + Support Vector Machine (SVM) with linear and radial kernel
    - Regression
    - 1-class classification
    - and multi-class classification
  + Random Forest (RF)
  + Gradient Boosting Machine (GBM) with Gaussian error

Both score1 and score2 were used to train the algorithms except for RF and GBM where only score1 was used.

Some of the algorithms are classification type and then don’t model score1 and score2 directly but their derivatives: “good” and “bad” answers dummies we mentioned earlier.

The algorithms were fitted on different data matrix built earlier to add diversity of fits.

In appendix, we provide a table summarizing for each individual model the response modelled, the algorithm type, the R package used and the data matrix on which the algorithms were trained.

Note that tuning parameters were not individually fine tuned per answer set but roughly set to obtain a good result globally. This could be improved to achieve a better performance per answer set

# Ensembles

The 5 folds cv holdout predictions from the previous models were blended using:

* at the second-level 2 algorithms
  + Regularized Generalized Linear Models with with Gaussian error using all individual models and some of their linear combinations
  + GAM with Gaussian error using a selection of linear combination of the individual models
* at the third-level Ordinary Least Square was used to blend the 2 previous ensembles

# Adjustments and rounding up

We found suboptimal to round up directly the predictions of our previous fits.

We chose to adjust our predictions before rouding up by using an optimization algorithm based on Nelder–Mead (default of the R function optim) to estimate the optimal adjustments.

We optimized two functions and selected the model with the best predictive accuracy:

SQWKappa(y,round(xx[1]+xx[2]\*x,0))

SQWKappa(y,round(xx[1]+xx[2]\*x+xx[3]\*x^2,0))

Where SQWKappa is a function to compute the competition metric, y the actual domain\_score, x the non-adjusted predictions and xx the coefficient to estimate.

# Final solution

We selected the best adjusted ensemble (among the 6 adjusted ensembles) using the 5 folds cv SQWKappa scores obtained for each adjusted ensemble.

For set2, the final solution was selected manually. This was done to ensure a distribution of the predictions for the validation set in line with the distribution of the holdout predictions obtained for the training set. The manual selection for set 2 improved slightly the public leaderboard score.

# Predictions for the test set

Predictions for the testing set were generated using the algorithms trained on the full data set.

# Code description

2 script files are provided.

Gxav TRAINING CODE.R = to train algorithms on the training set

Gxav TEST CODE.R = to predict scores for the test set

The following home-brew functions were developed

* **SQWKappa ()**
  + description: computes SQWKappa
  + input: predictions, actual values for one answer set
  + output: SQWKappa for one answer set
* **MQWKappa ()**
  + description: computes competition metric
  + input: vector of SQWKappa, weights per answer set
  + output: competition metric
* **Kappa\_score ()**
  + description: displays SQWKappa per answer set and competition metric
  + input: vector of predictions
  + output: SQWKappa per answer set and competition metric
* **Spear\_score ()**
  + description: computes Spearman rho per answer set
  + input: vector of predictions
  + output: spearman rho per answer set
* **Group()**
  + description: groups predictions per answer set
  + input: list of predictions per answer set
  + output: vector of predictions
* **graph1()**
  + description: visualizes effect of proxies
  + input: proxy
  + output: graph visualizing for each answer set the relation with actual score, the distribution of the proxy and the spearman rho
* **graph1b()**
  + description: visualizes relation with score
  + input: vector of predictions
  + output: graph visualizing for each answer set the relation with actual score, the distribution of predictions and the spearman rho
* **graph2()**
  + description: plots diagnostic graph of predictions vs actual values
  + input: vector of predictions, actual values
  + output: scatterplot
* **update\_split()**
  + description: updates the split per answer set of the data.frame “training”
  + input: none
  + output: updated training.Ex
* **RIDIT()**
  + description: computes ridit coding
  + input: vector used to produce the mapping between original data and ridit score, vector for which new ridit scores have to be computed
  + output: vector of ridit scores
* **RIDIT\_mat()**
  + description: repeats ridit coding for each column of a matrix
  + input: matrix used to produce the mapping between original data and ridit score, matrix for which new ridit scores have to be computed output: matrix of ridit scores
* **conc()**
  + description: concatenates text excluding the following words "the","a","an","this","that","these","those","to","is","are","be","or","and”
  + input: vector of words
  + output: text
* **stem\_text ()**
  + description: stems text
  + input: text
  + output: text of stemmed words
* **split\_k()**
  + description: produces 5 random subsets of row ids
  + input: seed, data to be split
  + output: list of row ids for 5 subsets
* **cv5\_GNET**()
  + description: executes the 5 folds cross-validation procedure of Regularized GLM fits with Gaussian error
  + input: data, list of 5 subsets of row\_id, response, elastic mixing parameter, logical value whether the data matrix need to be standardized, seed
  + output: list of objects representing the 6 fitted glmnet (5 folds cv+ full data) and 5-folds cv holdout prediction
* **useful\_words**()
  + description: provides indices of variables with non null coefficients from a regularized GLM
  + input: regularized GLM fit
  + output: list of indices for each round of the 5-folds cv procedure
* **conc\_remove ()**
  + description: concatenates text keeping only a list of words
  + input: vector of words, words to be kept
  + output: text
* **stem\_remove\_text ()**
  + description: stems text keeping only a list of words
  + input: text, , words to be keept
  + output: text of stemmed words
* **stem\_n\_grams ()**
  + description: computes for each answer set n-grams of stemmed words keeping only a list of words
  + input: answer set indice, words to be kept
  + output: list of n-grams of stemmed words for each round of the 5-folds cv procedure
* **shorten ()**
  + description: removes unfrequent terms (less than 5 times) from a list of terms for each round of the 5-folds cv procedure
  + input: list of terms for each round of the 5-folds cv procedure
  + output: list of reduced terms for each round of the 5-folds cv procedure
* **Create\_Feat\_mat ()**
  + description: creates features matrix for each round of the 5-folds cv procedure
  + input: list of terms for each round of the 5-folds cv procedure
  + output: list of features matrix for each round of the 5-folds cv procedure
* **Ass\_token ()**
  + description: assigns number to words (token) of a list of features matrix
  + input: list of features matrix for each round of the 5-folds cv procedure
  + output: list of features matrix with token for each round of the 5-folds cv procedure
* **Inc\_mat ()**
  + description: creates incidence document term matrix for each round of the 5-folds cv procedure
  + input: list of features matrix with token for each round of the 5-folds cv procedure
  + output: list of incidence matrix for each round of the 5-folds cv procedure
* **RIDIT\_mat5 ()**
  + description: scales with ridit coding a list of incidence matrix for each round of the 5-folds cv procedure
  + input: list of matrix used to produce the mapping between original data and ridit score, list of matrix for which new ridit scores have to be computed
  + output: list of scaled incidence matrix for each round of the 5-folds cv procedure
* **METRICS ()**
  + description: computes list of metrics (incl. delta tfidf, bns) (one vector per metric)
  + input: matrix, response, vector of row id
  + output: list of metrics
* **cv5\_METRICS ()**
  + description: computes list of metrics (incl. delta tfidf, bns) for each round of the 5-folds cv procedure
  + input: matrix, response, list of 5 subsets of row id
  + output: list of metrics for each round of the 5-folds cv procedure
* **cv5\_METRICS.2 ()**
  + description: computes list of metrics (incl. delta tfidf, bns) for each round of the 5-folds cv procedure
  + input: list of matrix for each round of the 5-folds cv procedure, response, list of 5 subsets of row id
  + output: list of metrics for each round of the 5-folds cv procedure
* **scal\_mat ()**
  + description: scales matrix
  + input: matrix, list of metrics (incl. delta tfidf, bns) for each round of the 5-folds cv procedure, indice of the scaling metric to be used
  + output: list of scaled matrix for each round of the 5-folds cv procedure
* **scal\_mat.2 ()**
  + description: scales matrix
  + input: list of matrix matrix, list of metrics (incl. delta tfidf, bns) for each round of the 5-folds cv procedure, indice of the scaling metric to be used
  + output: list of scaled matrix for each round of the 5-folds cv procedure
* **cv5\_GNET2**()
  + description: executes the 5 folds cross-validation procedure of Regularized GLM fits with Gaussian error
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, elastic mixing parameter, seed
  + output: list of objects representing the 6 fitted glmnet (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_SVM**()
  + description: executes the 5 folds cross-validation procedure of SVM for regression with linear kernel
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, cost, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_RAD2**()
  + description: executes the 5 folds cross-validation procedure of SVM for regression with radial kernel
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, cost, gamma, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_SVMc**()
  + description: executes the 5 folds cross-validation procedure of SVM for classification with linear kernel
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, cost, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_RAD2**()
  + description: executes the 5 folds cross-validation procedure of SVM for classification with radial kernel
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, cost, gamma, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_SVMm**()
  + description: executes the 5 folds cross-validation procedure of SVM for multinomial regression with linear kernel
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, cost, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_RADm2**()
  + description: executes the 5 folds cross-validation procedure of SVM for multinomial regression with radial kernel
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, cost, gamma, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_RAD** ()
  + description: executes the 5 folds cross-validation procedure of SVM for regression after having reduced the data matrix using PCA and metrics value under a threshold
  + input: list of data, list of metrics for each round of the 5-folds cv procedure, threshold for the metric value, list of 5 subsets of row\_id, response, cost, gamma, kernel, seed
  + output: list of objects representing the 6 fitted SVM (5 folds cv+ full data) and 5-folds cv holdout prediction
* **CVGNETcr**()
  + description: executes Regularized multinomial regression
  + input: data matrix, data matrix to use for predictions, response, elastic mixing parameter, logical value whether the data matrix need to be standardized, seed
  + output: objects representing the fitted glmnet cv holdout prediction and holdout predictions
* **Cv5\_GNETcr**()
  + description: executes the 5 folds cross-validation procedure of Regularized multinomial regression
  + input: list of data for each round of the 5-folds cv procedure, list of 5 subsets of row\_id, response, elastic mixing parameter, logical value whether the data matrix need to be standardized, seed
  + output: list of objects representing the 6 fitted glmnet (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_DATAG** ()
  + description: reduces matrix using only variables with non null coefficients from a regularized GLM
  + input: data to be reduced, regularized GLM fit
  + output: list of matrix for each round of the 5-folds cv procedure
* **cv5\_COMBGP** ()
  + description: combines : list of matrix for each round of the 5-folds cv procedure with a unique matrix
  + input: list of matrix for each round of the 5-folds cv procedure with a unique matrix
  + output: list of matrix for each round of the 5-folds cv procedure
* **cv5\_RF**()
  + description: executes the 5 folds cross-validation procedure of RF fits
  + input: list of data matrix for each round of the 5-folds cv procedure, list of 5 subsets of row id, response, indices of the predictors, the name of distribution to use, the maximum depth of variable interactions, the total number of trees to fit, the shrinkage parameter, the fraction of the training set observations randomly selected, nb of cv to perform, the seed
  + output: list of objects representing the 6 fitted RF (5 folds cv+ full data) and 5-folds cv holdout prediction
* **cv5\_GBM.step**()
  + description: executes the 5 folds cross-validation procedure of GBM fits with early stop
  + input: list of data matrix for each round of the 5-folds cv procedure, response, list of 5 subsets of row id, maximum nb of trees, the numbers of trees to add at each cycle, the maximum depth of variable interactions, the shrinkage parameter, the fraction of the training set observations randomly selected, seed
  + output: list of objects representing the 6 fitted gbms (5 folds cv+ full data) and 5-folds cv holdout prediction
* **Cv5\_GAM()**
  + description: executes the 5 folds cross-validation procedure of GAM fits
  + input: data, list of 5 subsets of row\_id, constant multiplier used to inflate the model degrees of freedom
  + output: list of objects representing the 6 fitted gams (5 folds cv+ full data) and 5-folds cv holdout prediction
* **Cv5\_GAM()**
  + description: executes the 5 folds cross-validation procedure of Ordinary Least Square fits
  + input: data, list of 5 subsets of row\_id
  + output: list of objects representing the 6 fitted ols (5 folds cv+ full data) and 5-folds cv holdout prediction
* **OPT1d()**
  + description: executes an optimization algorithm based on Nelder–Mead (default of the R function optim) to estimate the optimal coefficients for the function SQWKappa(y,round(xx[1]+xx[2]\*x,0))
  + input: vector of predictions, response, seed
  + output: coefficients
* **cv5\_OPT1d()**
  + description: executes the 5 folds cross-validation procedure of a optimization algorithm based on Nelder–Mead (default of the R function optim) to estimate the optimal coefficients for the function SQWKappa(y,round(xx[1]+xx[2]\*x,0))
  + input: vector of predictions, response, , list of 5 subsets of row\_id, seed
  + output: list of coefficients for each round of the 5-folds cv procedure and holdout predictions
* **OPT2d()**
  + description: executes an optimization algorithm based on Nelder–Mead (default of the R function optim) to estimate the optimal coefficients for the function SQWKappa(y,round(xx[1]+xx[2]\*x+ xx[3]\*x^2,0))
  + input: vector of predictions, response, seed
  + output: coefficients
* **cv5\_OPT2d()**
  + description: executes the 5 folds cross-validation procedure of a optimization algorithm based on Nelder–Mead (default of the R function optim) to estimate the optimal coefficients for the function SQWKappa(y,round(xx[1]+xx[2]\*x+xx[3]\*x^2,0))
  + input: vector of predictions, response, , list of 5 subsets of row\_id, seed
  + output: list of coefficients for each round of the 5-folds cv procedure and holdout predictions
* **range\_ADJ()**
  + description: cap predicted values range with possible score ranges
  + input: vector of predictions, score ranges for each answer set
  + output: adjusted predictions
* **Group\_t()**
  + description: groups predictions per answer set for test set
  + input: list of predictions per answer set
  + output: vector of predictions
* **graph3()**
  + description: vizualizes if predictions are consistent with 5 folds cv predictions
  + input: predictions for test set, predictions for training set
  + output: vector of predictions
* **update\_split\_t()**
  + description: updates the split per answer set of the data.frame “test”
  + input: none
  + output: updated data.frame
* **stem\_n\_grams\_t()**
  + description: computes for each answer set n-grams of stemmed words keeping only a list of words
  + input: answer set indice, words to be kept
  + output: n-grams of stemmed words
* **shorten\_t()**
  + description: removes unfrequent terms (less than 5 times) from a list of terms
  + input: list of terms
  + output: list of reduced terms
  + note: bug here, the reduction should have followed the one done for the training set. The impact is however insignificant.
* **Ass\_token\_t()**
  + description: assigns number to words (token) according to what was done for the training set
  + input: features matrix for testing set, features matrix for full training set
  + output: features matrix with token
* **Inc\_mat\_t()**
  + description: creates incidence document term matrix
  + input: features matrix with token for testing set, number of column of features matrix for full training set
  + output: incidence matrix
* **scal\_mat0()**
  + description: scales matrix using metrics computed for full training set
  + input: matrix, list of metrics (incl. delta tfidf, bns) for full training set, indice of the scaling metric to be used
  + output: scaled matrix
* **pred\_RAD()**
  + description: produces predictions for the model combining PCA and SVM
  + input: list of data matrix of training set, list of metrics (incl. delta tfidf, bns) of training set, threshold for the metric value, fit for full training set, data matrix of testing set
  + output: predictions
* **DATAG.0** ()
  + description: reduces matrix using only variables with non null coefficients from a regularized GLM
  + input: data to be reduced, regularized GLM fit of full training set
  + output: reduced matrix
* **COMBGP.0** ()
  + description: combines incidence matrix with proxies
  + input: incidence matrix, proxies
  + output: data matrix
* **conv\_ADJ1d**()
  + description: adjusts predictions
  + input: predictions, coefficients from cv5\_OPT1d (full training set)
  + output: adjusted predictions
* **conv\_ADJ2d**()
  + description: adjusts predictions
  + input: predictions, coefficients from cv5\_OPT2d (full training set)
  + output: adjusted predictions

Other functions used are predefined R functions.

# How to the generate the solution

1. Install R 2.15.1 (2012-06-22) -- "Roasted Marshmallows". Note previous versions could produce slightly different results because of some changes in the mgcv package
2. Open R
3. Store external data files, training set and test set files in a directory
4. Select as a R working directory the directory where the files were stored
5. Ensure all packages are installed
   1. SOAR # to manage memory usage
   2. Hmisc and MASS # for descriptive analysis
   3. Matrix # to handle matrix
   4. RTextTools # to stem words
   5. tau # to produce n-grams
   6. tm # to remove duplicated blanks and punctuation marks
   7. clim.pact # to convert to lower case
   8. glmnet # to train a generalized linear model via penalized maximum likelihood
   9. glmnetcr # to train a multinomial generalized linear model via penalized maximum likelihood
   10. e1071 # to train Support Vector Machine
   11. gbm # to train Gradient Boosting Machine
   12. randomForest # to train Random Forest
   13. mgcv # to train Generalized Additive Models
6. Execute Gxav TRAINING CODE.R
7. Execute Gxav TEST CODE.R
8. 1 csv file should be generated “Gxav\_Sol.csv” equal to the solution selected for the private leaderboard of the competition

**Memory issue in R**

To handle with the memory issue in R and share the same data in different R sessions, we used the package “SOAR” to store useful information on the hard disk.

Whilst opening a new R session, we recommend the user to first ensure that the working directly is correctly set, load the package SOAR and call the function Objects() to find stored object caches on the search path and list the objects stored in them.

# Comments

The solution was not fine-tuned per answer set (our original purpose was to build an automated solution which could fit all).

Then, better results could be achieved per answer set by setting customized tuning parameters and selecting relevant individual models. However, we believe the potential gain in predictive accuracy would be small and the work load high.

To reduce computing time, one could consider to remove RF models from the solution. Indeed, the 2 RF fits are slow to execute in comparison with other models and add low value to the predictive accuracy.

The computing time can be also reduced by parallelization.

# Appendix

Note that algorithms trained with score2 are displayed here. They use the same data matrix, only the response differ. Their name is equivalent except that they start by “S2\_”.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Data matrix | | | | | | | | | | | | |
| Object name | Response modeled | Algorithm | R package | scal\_mat1.Ex | scal.IB\_mat1.Ex | scal.mat1.2.Ex | scal.IB\_mat1.2.Ex | scal\_mat3.Ex | scal.IB\_mat3.Ex | scal.mat3.2.Ex | scal.IB\_mat3.2.Ex | Ridit\_mat.Ex | Ridit\_mat.1.Ex | Ridit\_mat.2.Ex | bin\_mat reduced | proxies |
| GNET1R | Score1 | Regularized GLM with elasticnet mixing parameter=0.5 and gaussian error | glmnet |  |  |  |  |  |  |  |  | X |  |  |  |  |
| GNET1R.1 |  |  |  |  |  |  |  |  |  | X |  |  |  |
| GNET1R.2 |  |  |  |  |  |  |  |  |  |  | X |  |  |
| GNETns1 | X |  |  |  |  |  |  |  |  |  |  |  |  |
| GNETns1.2 |  |  | X |  |  |  |  |  |  |  |  |  |  |
| GNETns1.IB |  | X |  |  |  |  |  |  |  |  |  |  |  |
| GNETns1.IB.2 |  |  |  | X |  |  |  |  |  |  |  |  |  |
| GNETns3 |  |  |  |  | X |  |  |  |  |  |  |  |  |
| GNETns3.2 |  |  |  |  |  |  | X |  |  |  |  |  |  |
| GNETns3.IB |  |  |  |  |  | X |  |  |  |  |  |  |  |
| GNETns3.IB.2 |  |  |  |  |  |  |  | X |  |  |  |  |  |
| GNETcr3 | Multinomial regression with elasticnet mixing param. =0.5 | glmnetcr |  |  |  |  | X |  |  |  |  |  |  |  |  |
| GNETcr3.2 |  |  |  |  |  |  | X |  |  |  |  |  |  |
| GNETcr3.IB |  |  |  |  |  | X |  |  |  |  |  |  |  |
| GNETcr3.IB.2 |  |  |  |  |  |  |  | X |  |  |  |  |  |
| SVM3 | SVM for regression with linear kernel | e1071 |  |  |  |  | X |  |  |  |  |  |  |  |  |
| SVM3.2 |  |  |  |  |  |  | X |  |  |  |  |  |  |
| SVM3.IB |  |  |  |  |  | X |  |  |  |  |  |  |  |
| SVM3.IB.2 |  |  |  |  |  |  |  | X |  |  |  |  |  |
| SVMc3.2 | isgood | SVM for classification with linear kernel |  |  |  |  | X |  |  |  |  |  |  |  |  |
| SVMc3.IB | isbad |  |  |  |  |  |  | X |  |  |  |  |  |  |
| SVMc3b | isgood |  |  |  |  |  | X |  |  |  |  |  |  |  |
| SVMc3b.IB.2 | isbad |  |  |  |  |  |  |  | X |  |  |  |  |  |
| SVMm3 | Score1 | SVM for multinomial regression with linear kernel |  |  |  |  | X |  |  |  |  |  |  |  |  |
| SVMm3.2 |  |  |  |  |  |  | X |  |  |  |  |  |  |
| SVMm3.IB |  |  |  |  |  | X |  |  |  |  |  |  |  |
| SVMm3.IB.2 |  |  |  |  |  |  |  | X |  |  |  |  |  |
| RAD | PCA + SVM for regression with radial kernel |  |  |  |  |  |  |  |  | X |  |  |  |  |
| RAD2 | SVM for regression with radial kernel |  |  |  |  | X |  |  |  |  |  |  |  |  |
| RAD2.2 |  |  |  |  |  |  | X |  |  |  |  |  |  |
| RAD2.IB |  |  |  |  |  | X |  |  |  |  |  |  |  |
| RAD2.IB.2 |  |  |  |  |  |  |  | X |  |  |  |  |  |
| RADc2 | isgood | SVM for classification with radial kernel |  |  |  |  | X |  |  |  |  |  |  |  |  |
| RADc2.2 | isbad |  |  |  |  |  |  | X |  |  |  |  |  |  |
| RADc2.IB | isgood |  |  |  |  |  | X |  |  |  |  |  |  |  |
| RADc2.IB.2 | isbad |  |  |  |  |  |  |  | X |  |  |  |  |  |
| RADm2 | Score1 | SVM for multinomial regression with radial kernel |  |  |  |  | X |  |  |  |  |  |  |  |  |
| RADm2.2 |  |  |  |  |  |  | X |  |  |  |  |  |  |
| RADm2.IB |  |  |  |  |  | X |  |  |  |  |  |  |  |
| RADm2.IB.2 |  |  |  |  |  |  |  | X |  |  |  |  |  |
| SEL\_RF1 | Random Forest for regression | randomForest |  |  |  |  |  |  |  |  |  |  |  | X | X |
| SEL\_RF1.2 |  |  |  |  |  |  |  |  |  |  |  | X | X |
| SEL\_GBM1.step | GBM with gaussian error | gbm |  |  |  |  |  |  |  |  |  |  |  | X | X |