FIT 5149 – Applied data analysis

ASSIGNMENT 2 – DATA ANALYSIS CHALLENGE

THI BICH NGOC HOANG - 28496337
THAT BAO THU TON - 28576322
JANNINA ONG – 28979877

Table of Contents

Summary

This document summarises the processes of developing features and models to predict labels of all records in the file “testing_docs.txt”. The report will list all R libraries used, the pre-processing steps, the feature selection steps and the method used for algorithm developments.

Also, it is a guideline for any further reproduction purposes.

1. Data Preparation

Data preparation is the first step in the process of model development. This step was implemented in the Python programming language. Both “training_docs.txt” and “testing_docs.txt” are preprocessed using this technique.

Python library used: re, nltk, pandas, multiprocessing, sklearn.feature_extraction.text

Python code file: “Document preprocessing.ipynb”

Output file: “corpus2.csv” and “test 2.csv”

Data preparation steps are recorded below.

- Remove stop words
- Remove character “TEXT” at the beginning of the content
- Tokenise words by using regular expression with pattern r"\w+(?:[-.@']\w+)*"
- Lemmatise words
- Remove a word if the length of this word is less than 3
- Concatenate all remaining tokens into “nsw_token”, which will be used for feature selection in next stages
- Produce clean data input files under the names “corpus2.csv” and “test 2.csv”

2. Features Selection

Library used: h2o (version 3.20.0.10)
Example output: w2v_e30_v200_w30_f0

In the beginning, we considered two directions for feature selection using TF-IDF and using word embedding. After testing several models, we realised the embedding feature outperforms TF-IDF in this prediction task. The library h2o with the function h2o.word2vec supported our feature selection in all models.

From the H20 documentation, we try the following h2o.word2vec setting to produce features.

- Epochs: Specifies the number of training iterations to run.
- vec_size: Specifies the size of word vectors.
- window_size: This specifies the size of the context window around a specific word.
- min_word_freq: Specifies an integer for the minimum word frequency. Word2vec will discard words that appear less than this number of times.

Reference: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/word2vec.html

We extracted the features used for one of our best models as an example of features selection as an H2O object named “w2v_e30_v200_w30_f0”. This object has the following setting:

- epochs = 30
- Vector size = 200
- Window size = 30
- min_word_freq = 0

3. Model selection

This section will describe the models that we used and assessed to ensemble the final result of the test set. In general, four types of model have been implemented: linear-based models, support vector machine models, Naïve Bayes and tree-based models. The best results will be generated from each model family and ensembled by a voting mechanism to achieve the final prediction. Furthermore, most of the models we used are from the h2o and liquidSVM packages because of their scalability, parallelisability, and computing resource optimisation. For detailed confusion matrix of each model, please refer to the appendix

3.1 Naive Bayes (NB)

Model description: The first simple model built for analysing the input features is the Naïve Bayes with 5-folds CV. From this model, we will test and discuss more complex models to improve the accuracy in the following sections.

Model description: H2O

Number of CV: 5

Model name	Output file	Word2vec setting	Train accuracy	CV accuracy
NB	NB.txt	epoch = 20 vector_size =300 window_size=15 min_word_freq = 10	66.2%	66.1%

Table 1: Naïve Bayes model summary

3.2 Generalised Linear Model (GLM)

Model description: In this model family, the multinomial family of the GLM algorithm of the h2o library is applied to 100% training data with 5-folds cross-validation. The average accuracy achieved is 75.8%, and the average MSE is 0.233.

Model description: H2O

Number of CV: 5

Model name	Output file	Word2vec setting	Train accuracy	CV accuracy
GLM	GLM.txt	epoch = 20 vector_size =300 window_size=15 min_word_freq = 10	77%	75.8%

Table 2: GLM model summary

3.3 Support Vector Machine (SVM)

Model description: liquidSVM, H2O

Number of CV: 5

Model description:

- Least squared selection-radial kernel-one versus all: auto, grid search, 5-folds CV.
- Hinge selection radial-radial kernel-one versus all: auto, grid search, 5-folds CV.
- Least squared selection-radial kernel-all versus all: cost = 0.01, gamma =3.1, grid search, 5-folds CV.

Hence, we will choose the least overfitted model with the best CV accuracy. The best one is the "least squared selection-radial kernel-all versus all" with the training accuracy of 85.94% for 5-folds cross-validation and testing accuracy of 77.15% on 0.25 of the dataset.

Model name	Output file	Word2vec setting	Train accuracy	CV accuracy
SVM1	SVM1.txt	epoch = 20 vector_size =150 window_size=30 min_word_freq = 0	95%	75.8%
SVM2	SVM2.txt	epoch = 30 vector_size =200 window_size=30 min_word_freq = 0	96%	76%
SVM3	SVM3.txt	epoch = 30 vector_size =200 window_size=30 min_word_freq = 0	85.94%	77.15%

Table 3: SVM model summary

3.4 Auto Machine Learning H2O

Besides the above-mentioned traditional methods, we also applied function automl of the h2o library to generate the leaderboard of best algorithms used for this dataset. Finally, we chose the stacked ensemble model and distributed random forest model to make predictions.
For more information, please follow the documentation link: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html

3.4.1 Stacked ensemble

Model description: "H2O's Stacked Ensemble method is a supervised ensemble machine learning algorithm that finds the optimal combination of a collection of prediction algorithms using a process called stacking. Like all supervised models in H2O, Stacked Ensemble supports regression, binary classification and multiclass classification." (stacked ensemble, 2018)

Library: H2O
Number of CV: 5
Model summary

Model id	Output file	Word2vec setting	Train accuracy	CV accuracy
SE	SE.txt	epoch = 20 vector_size =150 window_size=25 min_word_freq = 5	96%	76.1%

Table 4: Stacked ensemble summary

3.4.2 Distributed Random forest

Model description: "Distributed Random Forest (DRF) is a powerful classification and regression tool. When given a set of data, DRF generates a forest of classification or regression trees, rather than a single classification or regression tree. Each of these trees is a weak learner built on a subset of rows and columns. More trees will reduce the variance. Both classification and regression take the average prediction over all of their trees to make a final prediction, whether predicting for a class or numeric value." (drf, 2018)
Library: H2O
Number of CV: 5
Model summary

Model id	Output file	Word2vec setting	Train accuracy	CV accuracy
DRF1	DRF1.txt	epoch = 30 vector_size =200 window_size=30 min_word_freq = 5	73.2%	75.2%
DRF2	DRF2.txt	epoch = 20 vector_size =200 window_size=20 min_word_freq = 5	73.5%	75.1%

Table 5: Distributed random forest summary

3.5 Ensemble

Ensemble models can gain advantages by reducing the variances of each classifiers. We applied a weighted voting technique for the ensemble to produce the final results. Each model is assigned a different vote. A ranking is first determined by highest CV accuracy and less overfitting. We allocated the better model (i.e., the model with the higher rank) with more votes and gave the underperforming model less votes. Table 6 summarises the votes of all members as follows.

Model id	ranking accuracy	ranking overfit	ranking
SVM3	4	0	4
GLM	2	1	3
DRF1	1	1	2
SVM2	2	0	2
SE	2	0	2
NB	0	1	1
SVM1	1	0	1
DRF2	1	1	1

Table 6: Ensemble votes summary

The R script used to ensemble result was written in “final ensemble.R”. The final prediction is calculated by getting the prediction of each model and multiplying it by the number of votes for that specific model. The prediction with the highest frequency is chosen.

4. Predicting result

We used the output result of the ensemble as our final submission. Our submission includes the following files:

- “testing labels pred.txt”: final prediction
- “Ensemble member.zip”: prediction of each member of the ensemble
- “Rcode.zip”: R codes used for predicting results

Bibliography

drf. (2018, October 16). Retrieved October 2018, from docs.h2o.ai: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html

stacked ensemble. (2018, October 16). Retrieved October 2018, from docs.h2o.ai: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/stacked-ensembles.html

Appendix

1. NB confusion matrix (Model id - NB)

```
Cross-Validation Metrics Summary:
                              mean           sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid cv_5_valid
accuracy                  0.66109097 0.0020064623  0.6571563 0.66535616 0.66142726  0.6589899  0.6625254
err                         0.338909 0.0020064623  0.3428437 0.33464384 0.33857277 0.34101012  0.3374746
err_count                     7211.6    41.714745     7294.0     7150.0     7259.0     7211.0     7144.0
logloss                   6.7339315   0.07697875  6.8193655   6.566059    6.71484   6.880244  6.6891484
max_per_class_error        0.682075 0.0067200125   0.673028  0.6787037 0.69308174 0.67191285  0.6936488
mean_per_class_accuracy   0.6553237 0.0019063982  0.6511345  0.6592634 0.65634114   0.653954 0.65592533
mean_per_class_error     0.34467632 0.0019063982 0.34886548 0.34073663 0.34365886 0.34604597 0.34407467
mse                      0.32851192 0.0020712463 0.33200717 0.32438737  0.3286845 0.33134198 0.32613856
r2                       0.99259484  4.879458E-5  0.9925498   0.992703  0.9925712   0.992509  0.9926412
rmse                      0.5731539 0.0018074771 0.57620066 0.56955016 0.57331014  0.5756231 0.57108545
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
      c1  c10  c11  c12  c13 c14 c15 c16 c17 c18 c19  c2 c20 c21 c22 c23 c3  c4 c5 c6 c7 c8 c9  Error
c1  2905   21    1    1  288  40  16   3   5  36 227 162  39   4  91  26 12 283  1 90 38 12 16 0.3271
c10  289 3325    9    2   30  28   0 171   5  15   2  87  32   3 357  46 45 272  1  7 36 13 18 0.3063
c11   27   29 4458   16   11   6   0  24   2   0   1  40  19  60  68  58  8  38  1  1  8  4  0 0.0863
c12   28    8   84 5197    0   4   0  29   1   0   0   2   1   6  37  83  7  28  0  0  0  0  0 0.0577
c13  263    7    0    0 3119  55   1  10   0   0   3 277   2   0  58  29  5 172  2 37 16 14 10 0.2355
                Rate
c1  =    1,412 / 4,317
c10 =    1,468 / 4,793
c11 =      421 / 4,879
c12 =      318 / 5,515
c13 =      961 / 4,080

---
        c1  c10  c11  c12  c13  c14  c15  c16  c17  c18  c19   c2  c20  c21  c22  c23   c3   c4   c5   c6
c5      40   23   46    1    7    6    0   28    0    1    2   32    2   50   53   37    2   46 3939    0
c6     102   10    1    0   59  125  375    4  516   38  432   60   13    0  118   54   20  172    0 1963
c7     154   13    2    0   14  187    9    5   18   74   16  323   22   12   97   52   26  316    0   45
c8      13   16    0    0  119  178    3   21    5    2    0  513   10    0   20   48   38  105    1    3
c9      26   12    0    0    7  170   17    2  219  710    2  426   58    0   98   62  121  133    0   96
Totals 5829 4278 4964 5310 4627 6522 4802 1861 4949 4417 4804 5654 4656 4975 4034 3522 3349 5970 3975 3917
        c7   c8   c9  Error              Rate
c5       2    2    0 0.0880 =      380 / 4,319
c6      46   12  119 0.5369 =    2,276 / 4,239
c7    3233   23   73 0.3142 =    1,481 / 4,714
c8      78 2940  127 0.3066 =    1,300 / 4,240
c9      49  105 2881 0.4453 =    2,313 / 5,194
Totals 4697 4430 4854 0.3377 = 35,926 / 106,396
```

2. Generalised Linear Model confusion matrix (model id - GLM)

```
Cross-Validation Metrics Summary:
                              mean           sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid cv_5_valid
accuracy                  0.75791967  0.001997026  0.7532607  0.7564772  0.7599925   0.758577  0.7612909
err                       0.24208035  0.001997026 0.24673933 0.24352282 0.24000752 0.24142301 0.23870908
err_count                     5151.4     54.16198     5297.0     5132.0     5110.0     5144.0     5074.0
logloss                    0.7956744  0.007061446  0.8122392  0.8019663 0.78808665  0.7903327  0.7857472
max_per_class_error       0.61790025  0.009800818  0.6409736  0.6057907  0.6043956  0.6260965  0.6122449
mean_per_class_accuracy    0.7532143 0.0016269137 0.74977463 0.75229347 0.75435215  0.7529148 0.75673634
mean_per_class_error      0.24678572 0.0016269137 0.25022537  0.2477065 0.24564785 0.24708524 0.24326363
mse                       0.23326054 0.0018065226 0.23733422 0.23520674 0.23152643 0.23129325 0.23094209
null_deviance              133254.48     572.2195   134481.2  131953.58  133339.05  133436.28  133062.34
r2                        0.99474174 5.3076135E-5 0.99462724  0.9946806 0.99477434 0.99481386  0.9948128
residual_deviance           33863.36    369.34055  34874.305  33801.277  33558.305  33679.234  33403.684
rmse                      0.48296332 0.0018672153  0.4871696 0.48498118 0.48117194 0.48092955 0.48056436
```

```
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
      c1  c10  c11  c12  c13 c14 c15 c16 c17 c18 c19  c2 c20 c21 c22 c23  c3  c4 c5 c6 c7 c8 c9  Error
c1  2987   36    6    1  175  38  12  81   1  40 168 147  39  14  94  34 24 266  3 65 48 13 25 0.3081
c10   69 4003    5    5   16   5   1  90   6  14   5  27  26   1 268  34 19 131  5  5 18 13 27 0.1648
c11   10    1 4765   18    3   0   0   6   2   0   2   4   2  10  22  22  1   6  1  0  1  3  0 0.0234
c12    3    1    0 5479    0   0   0   4   1   0   0   0   0   1  11   8  0   6  1  0  0  0  0 0.0065
c13   96    6    1    0 3589  41   1  44   3   1   4 107   2   0  32  26  6  55  4 28 16 13  5 0.1203
                 Rate
c1  =    1,330 / 4,317
c10 =      790 / 4,793
c11 =      114 / 4,879
c12 =       36 / 5,515
c13 =      491 / 4,080

---
         c1  c10  c11  c12  c13  c14  c15  c16  c17  c18  c19   c2  c20  c21  c22  c23   c3   c4   c5   c6
c5        4    2    4   16    3    0    0    8    0    0    0    3    0    5   10    2    1   10 4248    1
c6       74    3    1    0   41   97  241   20  380   28  190   22   16    0  109   94   24  120    0 2620
c7       76   13    3    0   12   27    8   15    4   72   16  213   35    9   90   79   17  258    2   52
c8        9    7    1    0   41   71    9   42    6    4    4  495   15    1   15   10   51   69    1    6
c9       20   15    0    0    5   53   31    9   73  466    5  137   76    0   60   55   87  120    0   70
Totals 4558 4742 4908 5578 4378 4937 5531 3644 4870 4398 4881 4025 5056 5154 3992 4387 3283 5214 4313 3868
         c7   c8   c9  Error               Rate
c5        1    1    0 0.0164 =       71 / 4,319
c6       56    5   98 0.3819 =    1,619 / 4,239
c7     3585   41   87 0.2395 =    1,129 / 4,714
c8       78 3197  108 0.2460 =    1,043 / 4,240
c9       65   54 3793 0.2697 =    1,401 / 5,194
Totals 4884 4162 5633 0.2293 = 24,397 / 106,396
```

3. SVM Confusion matrix (Model id: SVM3)

```
Accuracy        0.771344787
Kappa           0.760817357
AccuracyLower   0.766249536
AccuracyUpper   0.7763807
AccuracyNull    0.052595962
AccuracyPValue  0
McnemarPValue   NA
```

4. Stacked ensemble Confusion matrix (Model id: SE)

```
5   5  0.999847
6   6  0.999883
7   7  0.999988
8   8  1.000000
9   9  1.000000
10 10  1.000000


H2OMultinomialMetrics: stackedensemble
** Reported on validation data. **

Validation Set Metrics:
=====================

MSE: (Extract with `h2o.mse`) 0.2368848
RMSE: (Extract with `h2o.rmse`) 0.4867081
Logloss: (Extract with `h2o.logloss`) 0.8087118
Mean Per-Class Error: 0.2433438
Null Deviance: (Extract with `h2o.nulldeviance`) 132739.1
Residual Deviance: (Extract with `h2o.residual_deviance`) 34281.29
AIC: (Extract with `h2o.aic`) NaN
Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,valid = TRUE)`)
========================================================================
Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
     c1 c10 c11  c12 c13 c14 c15 c16 c17 c18 c19 c2 c20 c21 c22 c23 c3 c4 c5 c6 c7 c8 c9  Error            Rate
c1  579  11   1    0  28   7   3  15   0  12  42 30   6   3  23   9  5 66  2 15  8  1  6 0.3360 =     293 / 872
c10  16 790   1    0   3   2   0  25   1   1   1  2   2   1  48   9  2 21  1  2  2  0  4 0.1542 =     144 / 934
c11   0   3 944    4   1   0   0   0   1   0   1  2   1   4   4   5  1  1  0  0  1  0  0 0.0298 =      29 / 973
c12   3   2   0 1091   0   0   0   1   0   0   0  0   0   0   3   0  0  2  0  0  0  0  0 0.0100 =    11 / 1,102
c13  26   3   0    0 758   7   0   7   0   0   0 30   0   0   7   3  3 14  1  6  4  0  1 0.1287 =     112 / 870

---
       c1 c10 c11  c12 c13 c14  c15 c16 c17 c18 c19  c2  c20  c21 c22 c23  c3  c4  c5  c6  c7  c8  c9  Error            Rate
c5      2   0   2    2   2   0    0   2   0   1   0   1    0    1   4   0   0   1 821   1   0   1   0 0.0238 =      20 / 841
c6     11   0   0    0   6  10   52   2  77   5  31   7    3    0  22  16   5  31   0 520  11   0  20 0.3727 =     309 / 829
c7     15   3   0    0   2   3    0   0   2  15   4  56    7    2  18  12   2  62   0  10 701   6  15 0.2503 =     234 / 935
c8      2   1   0    0  13  13    5  11   2   0   0 124    2    0   5   5   9  20   0   0  19 618  25 0.2929 =     256 / 874
c9      1   1   0    0   3  10    3   2  20 104   0  35   18    0  16  10  29  34   0  19  13  14 722 0.3150 =    332 / 1,054
Totals 890 938 965 1106 896 956 1073 783 956 893 922 879 1040 1012 791 842 618 1111 830 823 973 802 1096 0.2384 = 5,052 / 21,195

Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>,valid = TRUE)`
=======================================================================
Top-10 Hit Ratios:
    k hit_ratio
1   1  0.761642
2   2  0.880632
3   3  0.925360
4   4  0.947629
5   5  0.961972
6   6  0.970653
7   7  0.977495
8   8  0.982732
9   9  0.986742
```

5. Distributed random forest Confusion matrix (Model id: DRF2)

```
C:/Users/MSI-Guest/Desktop/totoro/
Extract cross-validation frame with `h2o.getFrame("automl_training_RTMP_sid_a31b_185")`
MSE: (Extract with `h2o.mse`) 0.3242822
RMSE: (Extract with `h2o.rmse`) 0.5694578
Logloss: (Extract with `h2o.logloss`) 1.090966
Mean Per-Class Error: 0.2530174
Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>,xval = TRUE)`
=======================================================================
Top-10 Hit Ratios:
    k hit_ratio
1   1  0.751247
2   2  0.872091
3   3  0.918722
4   4  0.942172
5   5  0.955940
6   6  0.964578
7   7  0.970141
8   8  0.973920
9   9  0.976690
10 10  0.978709


Cross-Validation Metrics Summary:
                             mean           sd cv_1_valid cv_2_valid cv_3_valid cv_4_valid cv_5_valid
accuracy                0.7512471 0.0027489597 0.74584824  0.7497653 0.75217134 0.75780517  0.7506455
err                    0.24875289 0.0027489597 0.25415176 0.25023475 0.24782863 0.24219483 0.24935447
err_count                  4238.8    46.892216     4331.0     4264.0     4223.0     4127.0     4249.0
logloss                 1.0909656 0.0018281353  1.0910288  1.0946928  1.0896266  1.0870264  1.0924537
max_per_class_error     0.6613941  0.009824923  0.6648794  0.6796247  0.6648794 0.63672924  0.6608579
mean_per_class_accuracy 0.74698263 0.0027233253 0.74138516   0.746008   0.747805 0.75337017   0.746345
mean_per_class_error    0.25301737 0.0027233253 0.25861487 0.25399205   0.252195 0.24662986 0.25365502
mse                     0.32428223 5.8778905E-4 0.32345188 0.32419628 0.32355633 0.32443893  0.3257677
r2                       0.9926798 1.3292017E-5  0.9926984 0.99268174 0.99269634  0.9926762 0.99264616
rmse                    0.5694573  5.158285E-4  0.5687283 0.56938237  0.5688201  0.5695954  0.5707606
>
```