Exercise 0 Dataset description

e12045110 Maria de Ronde
eXXXXXXXX Quentin Andre
e11921655 Fabian Holzberger

April 24, 2021

Classification Dataset: Email-Spam (link to dataset)

Bridges Dataset(link to dataset)

Kaggle: Amazon review

Dataset Description

The Amazon review dataset is a text dataset translated into vectors. There are 50 classes, which represent authors of different reviews. The goal of the classification is to predict the author of reviews. The dataset contains 750 instances with 100002 vectors, a nominal attribute representing a unique ID, 10000 numerical vectors and a nominal vector representing the class.
In the figure 2 one can see how many instances belong to each class. We can see that the dataset is unbalanced. As some classes have 20 instances where other classes have only 10 instances.
In tabel 1 one can see the distribution of the sum of the vectors. On average a word appears 309 times in all reviews, however there is a word which appears 187520 times, from 49 to 410 times in a review.

	0
count	10000.000000
mean	308.859700
std	2419.468303
min	0.000000
25%	9.000000
50%	21.000000
75%	220.000000
max	187520.000000

Figure 1: Distribution sum of vectors

Pre-Processing

There are no missing values in the dataset. The unique ID has been deleted from the data set as has no relevance to the class. The text vectors have values differentiating of a mean of 250 occurrences per instance to 1 occurrence in the entire dataset. With 250 occurrences per instance it appears as though stop words have not been deleted from the dataset, however as we do not have the raw data available we cannot be certain.
We sort the vectors from highest number of total occurrences to the smallest number of total occurrences. Words which only occur in one instance can be seen as unique identifier to one author. By sorting the data we enable, to train our model leaving some of the first and last columns out and test whether this improve the model.
In tabel X one can see the distribution of the sum of the vectors. On average a word appears 309 times in all reviews, however there is a word which appears 187520 times, from 49 to 410 times in a review.
In order to flatten the weight of words which occur more often in one instance we will also train the model on the natural logarithm of the original dataset. The dataset will be transferred into $ln(X) + 1$.

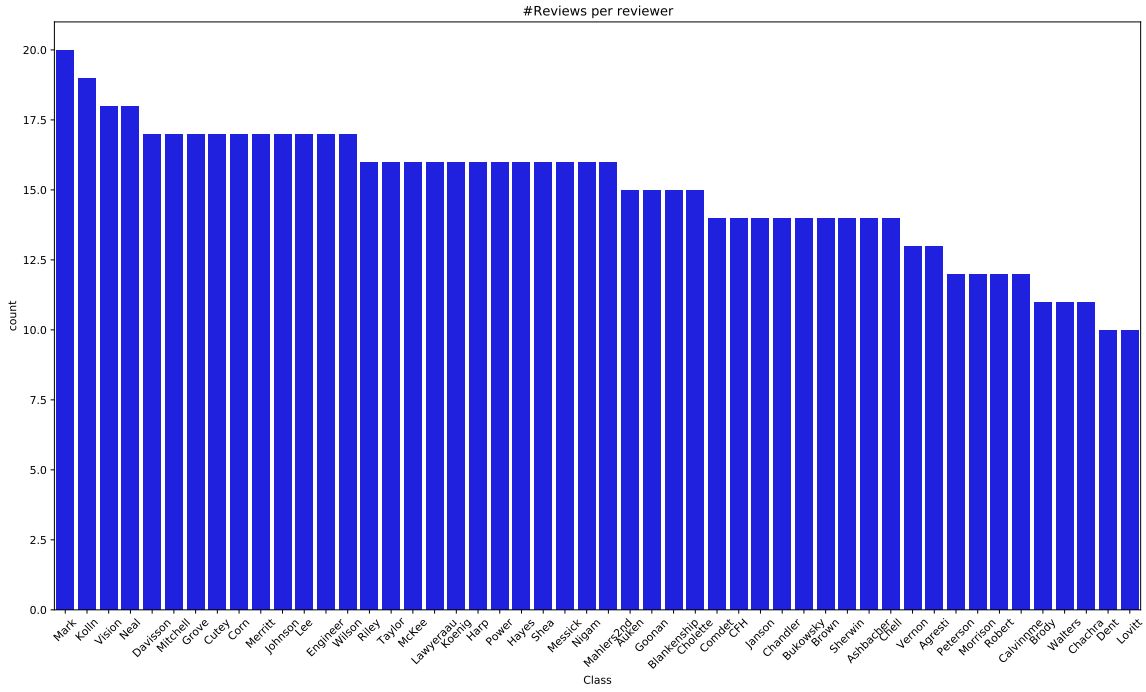

Figure 2: Instances per class

	Sum of vector
count	10000.000000
mean	308.859700
std	2419.468303
min	0.000000
25%	9.000000
50%	21.000000
75%	220.000000
max	187520.000000

	Scenario	Scenario description
0	10000	Select all 10000 attributes
1	8000	select the first 8000 attributes
2	6000	select the first 6000 attributes
3	50:10000	select the 50th till the 10000th attribute
4	50:8000	select the 50th till the 8000th attribute
5	50:6000	select the 50th till the 6000th attribute
6	100:10000	select the 100th till the 10000th attribute
7	100:8000	select the 100th till the 8000th attribute
8	100:6000	select the 100th till the 6000th attribute

We create 9 scenarios, which are shown in table X for which we train the three classifiers: Perceptron, Random forest and Naive Bayes with the multi-nominal distribution.
There are no missing values in the dataset. The unique IF has been deleted from the dataset as this has no

relevance to the class. The model might find a relation, which will not hold for the new instances. The text vectors have values differentiating from 250 occurences per instance to 1 occurence in the entire dataset. With 250 occurrences per instance it appears that stopwords have not been deleted from the dataset, however as the raw data is not available we cannot be certain.
In order to leave specific columns out of the the model, we sort the vectors. Vectors with the highest occurrences first and vectors with the least occurrences last.We will run the models leaving out a different number of vectors at the beginning and the end of the dataset.
In order to flatten the weight of words with high occurrences we will also train the model by taking the natural logarithm of the original vectors. The dataset is transferred into $ln(X) + 1$.
We create 9 different scenarios leaving out different number of vectors at the beginning and the end. The different scenarios can be found in table 3. We will run the three classifiers, perceptron, random forest and naive bayes, with a multinominal distribution, for all scenarios on the original dataset and on the logarithmic dataset, $ln(X) + 1$.

	Scenario	Scenario description
0	10000	Select all 10000 attributes
1	8000	select the first 8000 attributes
2	6000	select the first 6000 attributes
3	50:10000	select the 50th till the 10000th attribute
4	50:8000	select the 50th till the 8000th attribute
5	50:6000	select the 50th till the 6000th attribute
6	100:10000	select the 100th till the 10000th attribute
7	100:8000	select the 100th till the 8000th attribute
8	100:6000	select the 100th till the 6000th attribute

Figure 3: Scenarios

Parameter-Tuning

First we will split the data in two parts. One part for training and testing the model with different parameters and one part to validate the models afterwards. To extract the validation set, the hold out strategy has been used.
To split the rest of the data in a training dataset and a testing dataset, both, the hold out strategy and the cross validation using Xfold have been used. The results of the model on the test sets are used to tune the parameters on the model.

Perceptron

We determine the base parameters for the perceptron, alpha as 0.0005, eta as 1 penalty='none' and the max iterations are equal to 100.
The results of the 9 scenarios can be found in figures 4 and 5 using the hold out and cross validation strategy respectively. All train set of the $ln(X) + 1$ have an accuracy of a 100, for X, we could not obtain a perfect score when high occurring words were deleted. Comparing the accuracy of the test set of X to $ln(X) + 1$,we see that the logarithmic dataset performs better when we do not delete the words with high occurrences. After removing the top 100 words the accuracy of the X dataset is higher than the$ln(X) + 1$ dataset. Transforming the data into logarithmic values has the highest effect on the words which occur most often.

A clear difference can be seen between the performance of the hold out and the cross validation runs. The cross validation run clearly out performs the hold out run. Due to the high amount of classes, 50, and the relative low amount of instances it can happen that classes are not represented in the test set, some classes only appear 10 times in the entire dataset. The accuracy is sensitive to the chosen test and train set. With using the cross validation the entire dataset is used (except the part left out for validation) and the influence of chance decreases. In order to tune the hyper parameters we will use the cross validation and look at the results of the test set.Due to the high complexity of the data, and a relative little instances the classifiers can train the model to fit training data exactly. Therefore the results on the training data cannot be compared.
For the parameter tuning of on the perceptron classifier scenario 4 of the untransformed dataset is chosen. Even though, scenario 3 performed slightly better. The vectors with very low occurrences might have unique identifiers to classes, which increases over fitting.

	Scenario	Basic test	Basic train	LN test	LN train
0	10000	37.777778	91.666667	40.000000	100.0
1	8000	39.259259	91.296296	41.481481	100.0
2	6000	40.000000	88.148148	47.407407	100.0
3	50:10000	40.740741	100.000000	34.074074	100.0
4	50:8000	40.740741	100.000000	36.296296	100.0
5	50:6000	40.000000	100.000000	41.481481	100.0
6	100:10000	40.000000	100.000000	43.703704	100.0
7	100:8000	40.000000	100.000000	42.962963	100.0
8	100:6000	37.037037	100.000000	41.481481	100.0

Figure 4: Results scenario run perceptron, hold out

	Scenario	Basic test	Basic train	LN test	LN train
0	10000	40.133333	93.229630	50.133333	100.0
1	8000	40.000000	93.585185	50.266667	100.0
2	6000	39.466667	92.340741	51.200000	100.0
3	50:10000	51.600000	100.000000	50.666667	100.0
4	50:8000	51.200000	100.000000	50.266667	100.0
5	50:6000	49.066667	100.000000	50.400000	100.0
6	100:10000	50.800000	100.000000	49.333333	100.0
7	100:8000	50.533333	100.000000	50.133333	100.0
8	100:6000	50.266667	100.000000	48.933333	100.0

Figure 5: Results scenario run perceptron, cross validation

The first parameter we tune is the maximum number of iteration max_iter. We test for values $max\_iter \in \{2, 3, 4, 5, 10, 50, 100\}$. It can be seen that after 10 iterations the training set has a perfect fit, and the test set has the highest accuracy, as well as precision and recall. Therefore, we take $max\_iter = 10$ for the next parameter test.
Following the model has been ran for different learning rates ($eta0$), where $eta0 \in \{1^{-4}, 1^{-3}, 0.01, 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 1\}$. The model is indifferent for the learning rate and always obtains the same accuracy, precision and recall values. This means that the same weights are obtained.
Since the model is over fitting a regularization term has been introduced. At first α was set to be

$\in \{1e^{-4}, 1e^{-3}, 0.01, 0.5, 1\}$. The accuracy for both the training set as well as the test set dropped rapidly after 1^{-3}. Therefore, a second run with $\alpha \in \{1e^{-5}, 1e^{-4}, 0.0002, 0.0004, 0.0006, 0.0008, 1e{-}3\}$ has been done. For $\alpha = 0.0002$, the test set performs best with 49.16%.

	Alpha	Basic test	Basic train
0	0.00001	47.258560	100.000000
1	0.00010	47.532924	99.983526
2	0.00020	49.163740	99.473169
3	0.00040	41.922739	98.616470
4	0.00060	39.078139	94.471517
5	0.00080	41.027217	96.558869
6	0.00100	42.811677	96.806311

Figure 6: Accuracy perceptron alpha

Finally we run our final model with the following parameter settings, maximum number of iterations is 10, $eta0 = 1$, the regularization term is l1, with an α of 0.0002 against the validation set. This results in an accuracy of 50.4%. In the confusion matrix we can see that some names are never predicted and some names are predicted too often, like Brown.

Random forest

The second classifier is the random forest classifier. First it is checked which data pre-process steps will be applied to the model. The base parameter settings are the following: $ccp\alpha = 0$
$maxleaf = 50$
$maxdepth = 10$
$numberofestimators = 100$
The results of the different scenarios for hold out and cross validation can be found in the figures 8 and 8 respectively. The best scenario is scenario 1 on the natural logarithmic dataset with an accuracy of 49.17% in the cross validation. This model is less sensitive for the extreme values of words with an high occurrence as the results of $X\%$ and $LN(X) + 1$ are closer to each other, as well as the model taking the first attributes into account scoring highest on accuracy. Using cross validation the $LN(X) + 1$ always out performs the dataset X, where in the hold out run there is not pattern to be seen.
The following hyper parameters have been tested: number of estimators, the maximum depth, maximum leafs and the complexity cost parameter, ccp alpha. Starting with the number of estimators,the number of trees, given n estimators $\in \{1, 5, 10, 50, 70, 100, 200\}$.
Both the train set and the test set achieve an higher accuracy whenever the number of estimators increases. The number of estimators is set to 200. The second hyper parameter is the maximum number of leafs per branch, we test $maxleafs \in \{5, 10, 20, 50, 100\}$ The accuracy increases when the number of leafs increase to 50, with 100 the accuracy, precision and recall do increase for the training set, but not for the test set. The maximum number of leafs is set to 50.
After the number of trees and the leafs per branch, the depth of each tree is determined. where $maxdepth \in \{2, 3, 4, 5, 8, 10, 50, 100\}$. The accuracy for the training set is highest with a maximum depth of 10. The accuracy for the test set is highest with a maximum depth of 50 or a 100, the same scores are obtained. The maximum depth is set to 50.
Finally we will include minimal cost complexity pruning. We test for $ccp\alpha \in \{1e-5, 2e-5, 1e-4, 2e-4, 1e-3, 5e-3, 75e-4, 1e-2\}$. It can be seen that for $ccp\alpha < 0.005$ no pruning has been done. However, for

$ccp\alpha > 0.005$ both the train and the test have reduced scores. Therefor, we will set $ccp\alpha = 0.005$.

	Scenario	Basic test	Basic train	LN test	LN train
0	10000	42.222222	99.814815	43.703704	99.814815
1	8000	39.259259	99.814815	39.259259	99.814815
2	6000	39.259259	99.814815	38.518519	99.814815
3	50:10000	37.037037	99.814815	37.777778	99.814815
4	50:8000	37.777778	100.000000	36.296296	100.000000
5	50:6000	41.481481	99.814815	38.518519	99.814815
6	100:10000	37.037037	99.629630	36.296296	99.629630
7	100:8000	35.555556	99.444444	34.814815	99.444444
8	100:6000	37.037037	99.444444	37.777778	99.444444

Figure 7: Accuracy random forest hold out

	Scenario	Basic test	Basic train	LN test	LN train
0	10000	46.222564	99.703812	46.808604	99.720259
1	8000	48.432836	99.703731	49.174715	99.736652
2	6000	46.804214	99.868340	47.098332	99.851892
3	50:10000	47.245391	99.325143	47.396839	99.341618
4	50:8000	45.037313	99.489644	45.041703	99.506145
5	50:6000	46.661545	99.720286	47.695347	99.720286
6	100:10000	45.610184	99.193429	46.341089	99.176954
7	100:8000	46.501317	99.226215	46.652766	99.226215
8	100:6000	44.721247	99.374404	44.725637	99.374404

Figure 8: Accuracy random forest hold out

The final model is now tested on the validation data with the following parameter settings: $ccp\alpha = 0.005$
$max_leaf = 50$
$max_depth = 50$
$nestimators = 200$
This gives a accuracy of 58.66%. In the confusion matrix it can be seen that there are 7 labels not represented in the validation set. Six of the seven authors was also never predicted, one author was predicted 4 times.

Naive Bayes

The third classifier is the multinominal Naive Bayes classifier. For the base run we set α equal to 1. The nine scenarios are run for both hold out and cross validation. A clear pattern can be seen in the result. The accuracy of the original datset out performs the logarithmic dataset. Furthermore it can be seen that including less vectors improves the model, both the first and the last vectors from the dataset. Because the best results are obtained in scenario 8 additional scenarios are ran excluding additional vectors from the beginning and the end.
First 5 additional scenarios are created excluding more vectors in the end. The best results are obtained with scenario 11, and 12. Another 10 scenarios are considered leaving the first 150, 200, 250, 300 and 350 vectors out

until the 4000th and the 4500th vector. The accuracy can be found in table 10. the best accuracy is obtained in scenario 250 closely followed by scenario 22.

	Scenario	Scenario description
9	100:5500	select the 100th till the 5500th attribute
10	100:5000	select the 100th till the 5000th attribute
11	100:4500	select the 100th till the 4500th attribute
12	100:4000	select the 100th till the 4000th attribute
19	100:3500	select the 100th till the 3500th attribute

Figure 9: Additional Scenarios

	Scenario	Basic test	Basic train
20	150:4500	59.995610	99.884760
21	200:4500	59.995610	99.950631
22	250:4500	60.006585	99.967105
23	300:4500	59.411765	99.983553
24	350:4500	59.253731	99.983553
25	150:4000	60.140474	99.851838
26	200:4000	59.846356	99.901262
27	250:4000	59.997805	99.901262
28	300:4000	58.812555	99.967078
29	350:4000	59.407375	99.983553

Figure 10: Accuracy scenarios 20-29

The first 9 scenarios have been ran for different smoothing priors. We ran the model for alpha's $\in \{1e{-}3, 1e{-}2, 1e{-}1, 0.5, 1, 10, 100, 1000\}$. All results are equal for all alphas.

Performance-Analysis