Personality Trait Classifier from Natural Language

Guillem Bagaria i Portet

July 3, 2018

Contents

Chapter 1

Initial data exploration

1.1 The data

About the data:

The used dataset consists of 10,000 transcripts extracted from video clips of people talking directly into the camera. Each transcript has an assigned label for each one one of the 6 labels (extraversion', 'neuroticism', 'agreeableness', 'conscientiousness', 'openness' and 'interview'). The first 5 are from the Big Five personality traits, originally developed in 1961 in a US Air force paper, very commonly cited in psychological literature [1].

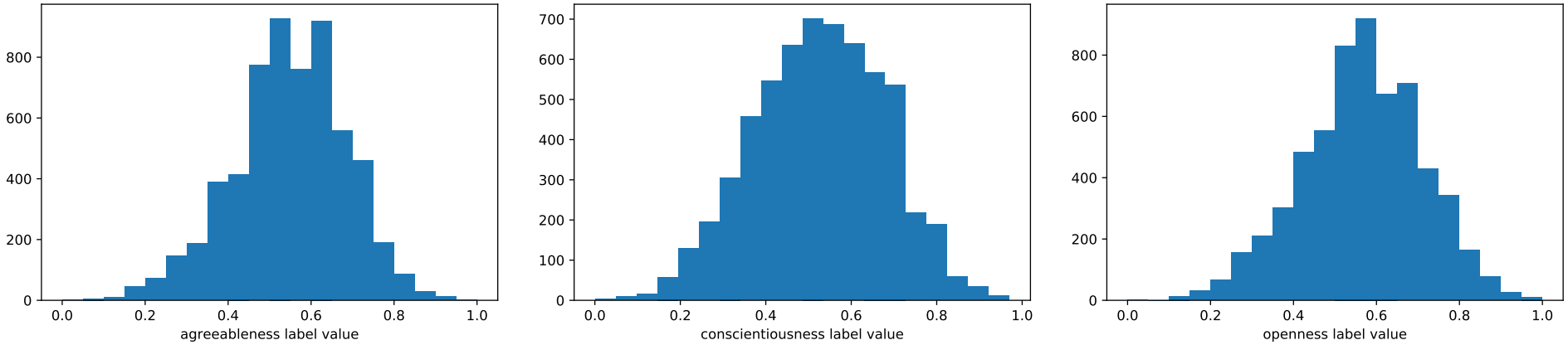
In order to estimate the ground truth, human perception while seeing the videos was used through Amazon Mechanical Turk [3]. Each rater had to rank a small batch of paired videos by comparing them, and point out which one was more extroverted, neurotic, interviewable, etc. All 10,000 videos were pairwise compared 321,684 times producing, for each label, a ranking between 0 and 1. The values are centered relatively close to 0.5 except for openness and agreeableness which are both centered closer to 0.6.

Because the labels are treated as a partial class membership (as opposed to multiple membership, which would imply that only one if them is actually true), in the *ChaLearn LAP 2016* paper they use an accuracy measure as an average distance from the ground truth for each personality trait. The best way to do this is to calculate the performance as some kind of distance from the ground truth value.

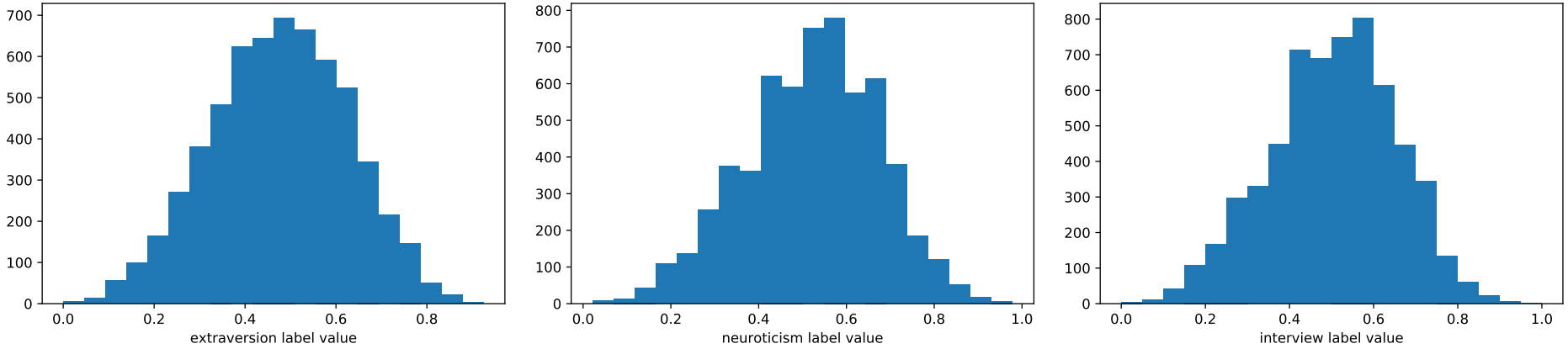
If we instead tried to assess the dominant trait, interpreting the ground truth and predicted labels as a multiple membership problem, it would be necessary to generate a label defined as such: $t = 1$ if the highest ranked trait corresponds to the highest ranked trait in the ground truth, $t = 0$ otherwise. However, even before trying this, I believe this reduces drastically the amount of data that might otherwise be valuable to for a learning algorithm. Similarly to the case of ranking the "interview" labels and setting an arbitrary threshold.

About the *interview* label:

The values for the interview label are 0.504 mean and 0.514 median and range from 0.0 to 1.0, the higher, the more inteviewable that person is. The job in question is not determined, so the labels are given for personality traits reasons. However, this might be a naïve understanding, because there might be significant bias towards other parameters that are not counted in the labels,



(a) agreeableness (b) conscientiousness (c) openness



(d) extraversion (e) neuroticism (f) interview

Figure 1.1: Distribution of the jobScreening data label values

such as gender or age. Biases should be lower (albeit not nonexistent) when asking for the more "objective" personality traits listed above.

The way the data is collected suggests that the personality traits are related to the *interview* label value. In order to better understand the predictions I would first use a simple linear regression model to find how the personality traits predict the *interview* label. The expectation is that this will

1. point out the factors (parameters) that determine interviewability and
2. isolate hypothetical bias in the *interview* label.

1.2 Fleiss's kappa coefficient (κ)

Fleiss's kappa coefficient (κ), as described in [5] is a statistic which measures inter-rater agreement in a many rater scenario, for qualitative (categorical) ratings that takes into account the possibility of agreement occurring by chance. In the case of the labels used in this dataset, we can use this as a threshold to reject labels that are deemed inconclusive. This fact adds to the problem of low contrast labels for the classifier algorithm during the training.

This coefficient can be calculated as

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

where p_0 is the accuracy among raters, and p_e is the hypothetical probability of agreeing by chance, calculated as

$$p_e = \sum_{j=1}^{k} p_j^2$$

whith

$$p_j = \frac{1}{N^2} \sum_{i=1}^{N} n_{ij}$$

where N is the total number of raters, n the number of ratings per rater such that n_{ij} and k the number of categories, in this case $k = 2$.

If all raters agree, $\kappa = 1$; if there is no agreement, aside from that occurring by chance, $\kappa = 0$; if there is less agreement than what would occur by chance, $\kappa < 0$.

We do not have the data from the ground truth estimation, however it is a reasonable hypothesis that the ambiguous values (close to the mean) are caused by higher disagreement between raters, as described by the Bradley–Terry model [6], used in the ranking of the clips.

1.3 Interview label prediction

The analysis and information extraction of the transcripts is the first step. In order to do this the Job Screening data is used to create a binary classifier using the automatic transcripts as data. This algorithm will load the natural language transcript data, divide the data into tokens (one or several words), remove unwanted tokens and other unnecessary characters, convert it into a TFIDF matrix in order to extract the relevant words. This will later improve the performance of the classifier algorithm.

The used algorithms to be compared will be a Support Vector Classifier with radial kernel and a Naive Bayes Classifier.

1.4 Threshold for dichotomic labels

In order to train a classifier model, categorical labels are required. As previously seen, the *interview* label follows a bell curve distribution between 0 and 1. The objective is to dichotomize the label into two categories (0 and 1) in order to feed the Support Vector Classifier.

Since many documents have a somewhat ambiguous label (that is, close to the mean label value, or ~0.5), the threshold for the training process will be centered in order to provide the classifier algorithm with better contrast. The center is set on the median value in order to correct for label imbalances in the initial threshold ranges.

In order to maximize the the classifier performance it's necessary to find the optimal standard deviation distance away from the mean as the threshold to discard from the training and test sets. Initially, the same standard deviation from the median value from both train and test data was discarded. However, a second experiment was conducted with several fixed values for the test set. In Figure 1.2 are the results for the first case, with identical thresholds applied to both testing an training data. The reasons to do this are not exclusively to make the classifier's job easier, but also to filter out disagreements from the raters, as discussed in the previous section.

In Figure 1.2, the standard deviation value as discard threshold of both train *and* test that maximizes the ROC Score. The maximum is found at around Sigma = 1.72. The posterior results are decreasing until they start behaving erratically due to small test dataset and overfitting due to the small training set. However, the steady increase in performance is caused both by the increased contrast between labeled documents as well as the decrease in testing sample.

From the initial results, the following step was to perform the same tests with fixed testing sets. In order to find the optimal test set threshold, several cuts from the test set will be made and subsequently a training set threshold sweep will be calculated for each one. Since decreasing the testing set size directly translates into an improved score, what will be observed is the average slope of the training set sweep. The results can be seen in Figure 1.3 and Figure 1.4.

The ROC score against the training sigma threshold for each value of removed sigma in the test set can be seen in Figure1.4 The training set optimal sigma is very stable, at around 0.278. The rate at which the ROC score decreases with training sigma is steeper as the test sigma increases until it reaches 1 sigma. At this point the increase in training sigma has little impact on the ROC suggesting that the dominant factor in the ROC score becomes the small test sample size. The optimal value for which the SVC classifier obtains the best results, given the training and test data are a testing threshold of 0.8 sigma and a training threshold of 0.278 sigma.

The plots can be seen in Figure 1.5 for the Support Vector Classifier and in Figure 1.6 for the Naive Bayes Multinomial classifier.

The sigma parameters together with a Naive Bayes Multinomial classifier produce a stable optimal value, but the ROC score decreases some decimal points compared to an rbf kernel SVC.

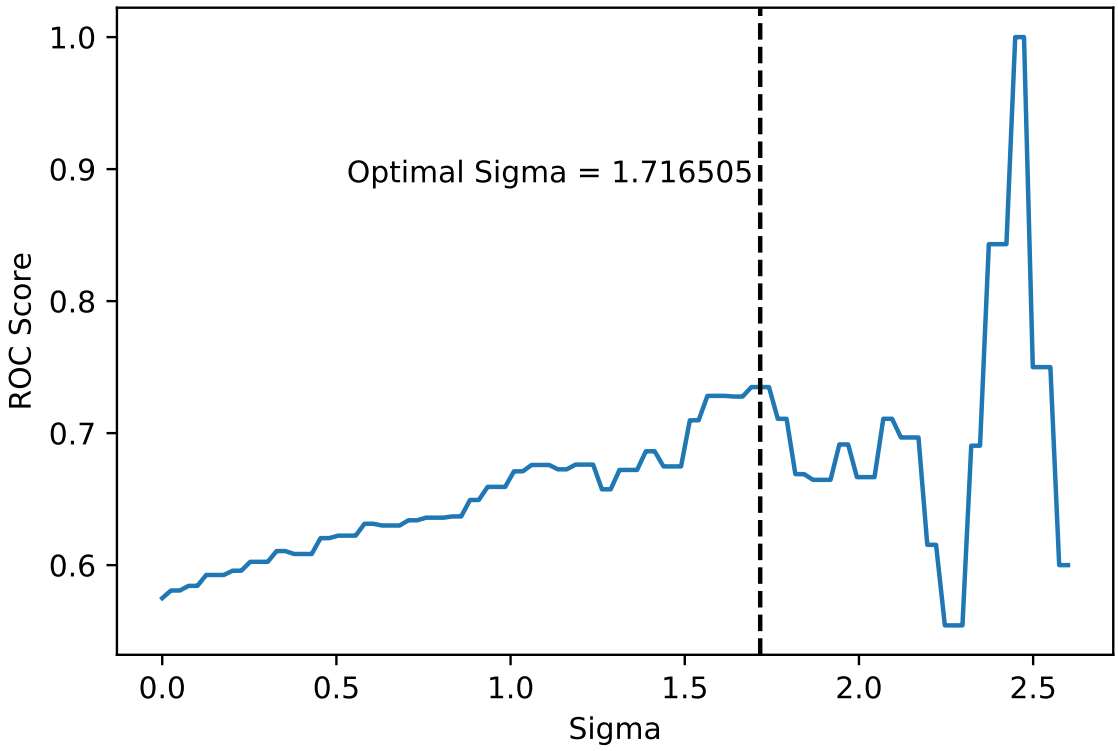
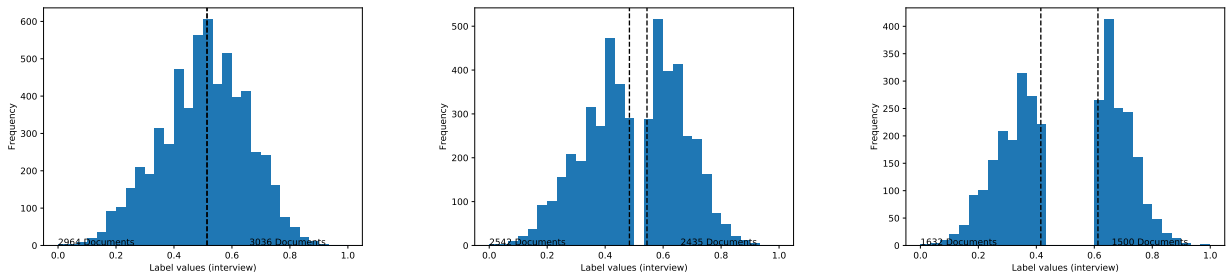
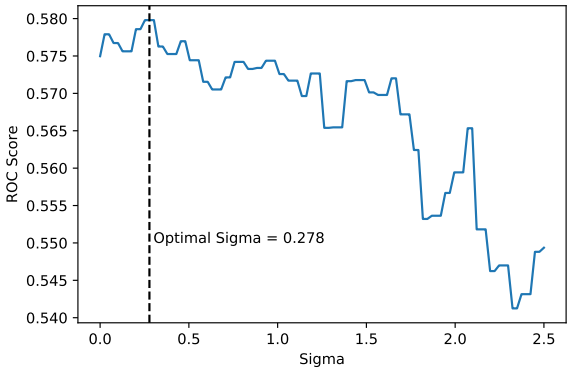


Figure 1.2: Optimal sigma for thresholds applied at both train and test splits.

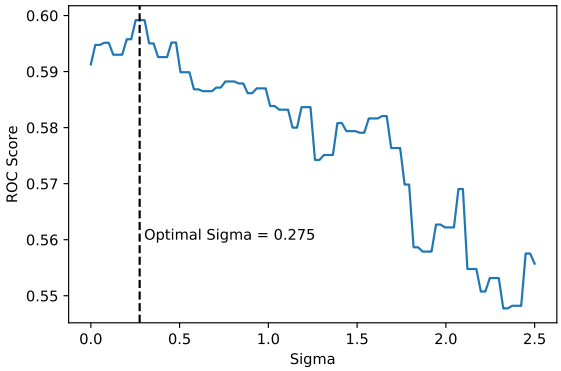


(a) 0 sigma (b) 0.202 sigma (c) 0.656 sigma

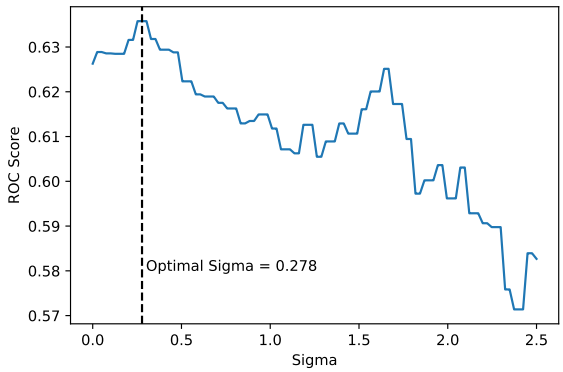
Figure 1.3: Distribution of the jobScreening interview label values after removing increasingly wide central values.



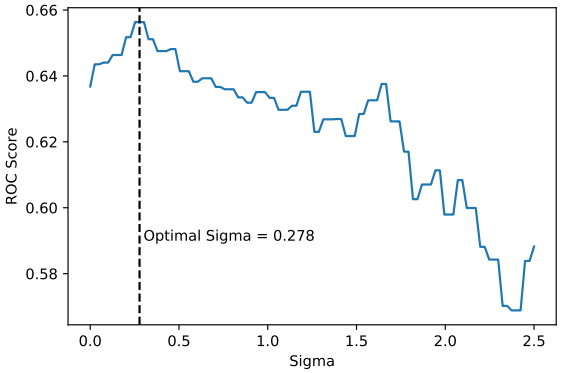
(a) Complete test set. The score varies very little as the training sigma increases because the test set includes too many ambiguous documents.



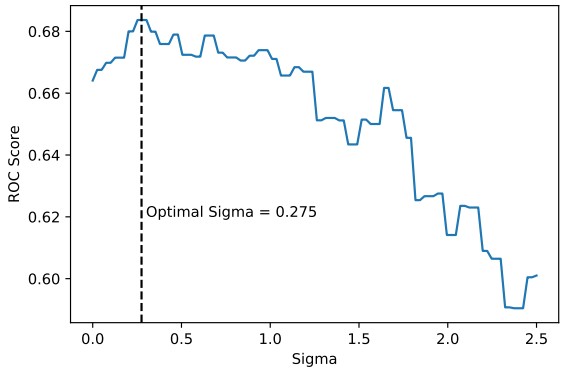
(b) Test set without 0.2 Sigma. The slope to and from the optimal are steep.



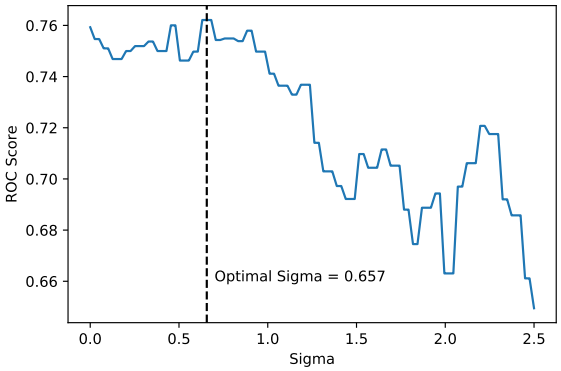
(c) Test set without 0.5 Sigma. This case shows a steep slope after the optimum.



(d) Test set without 0.8 Sigma. This case shows a steep slope both after and before the optimum.



(e) Test set without 1.0 Sigma. The score plateaus although the training optimal is still consistent with all previous test sets.



(f) Test set without 1.5 Sigma. The score shows the same plateau shape, but behaves increasingly erratically, first signs of small test set.

Figure 1.4

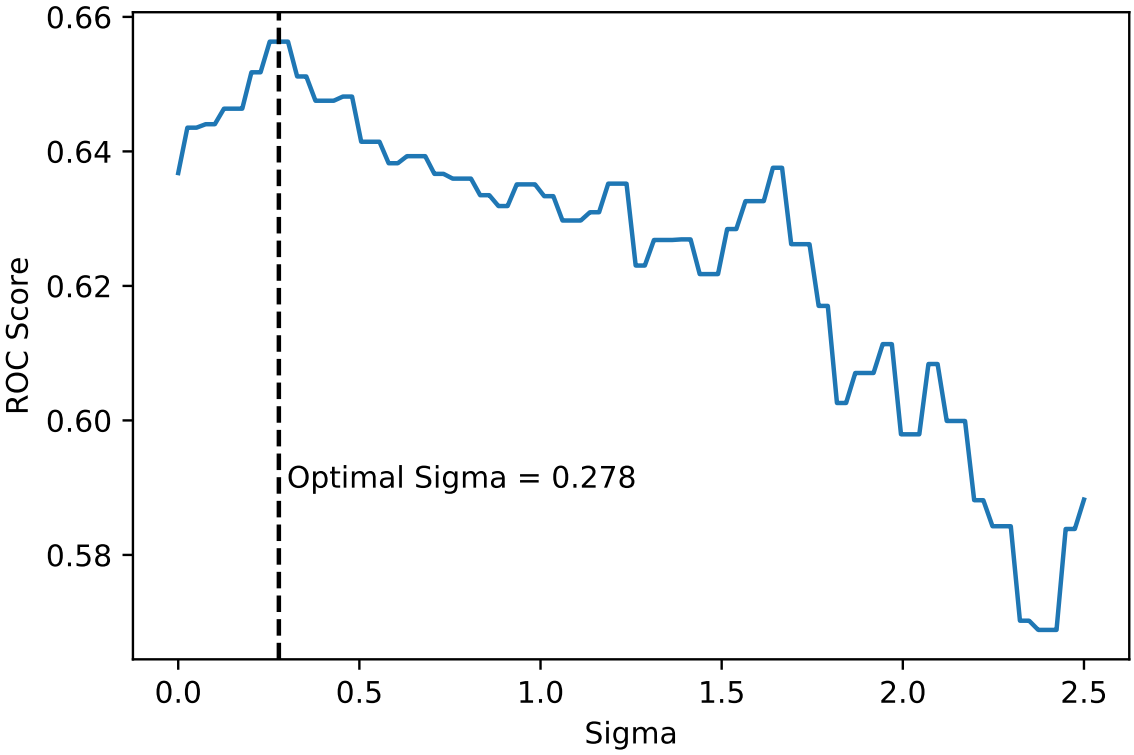


Figure 1.5: rbf kernel SVC

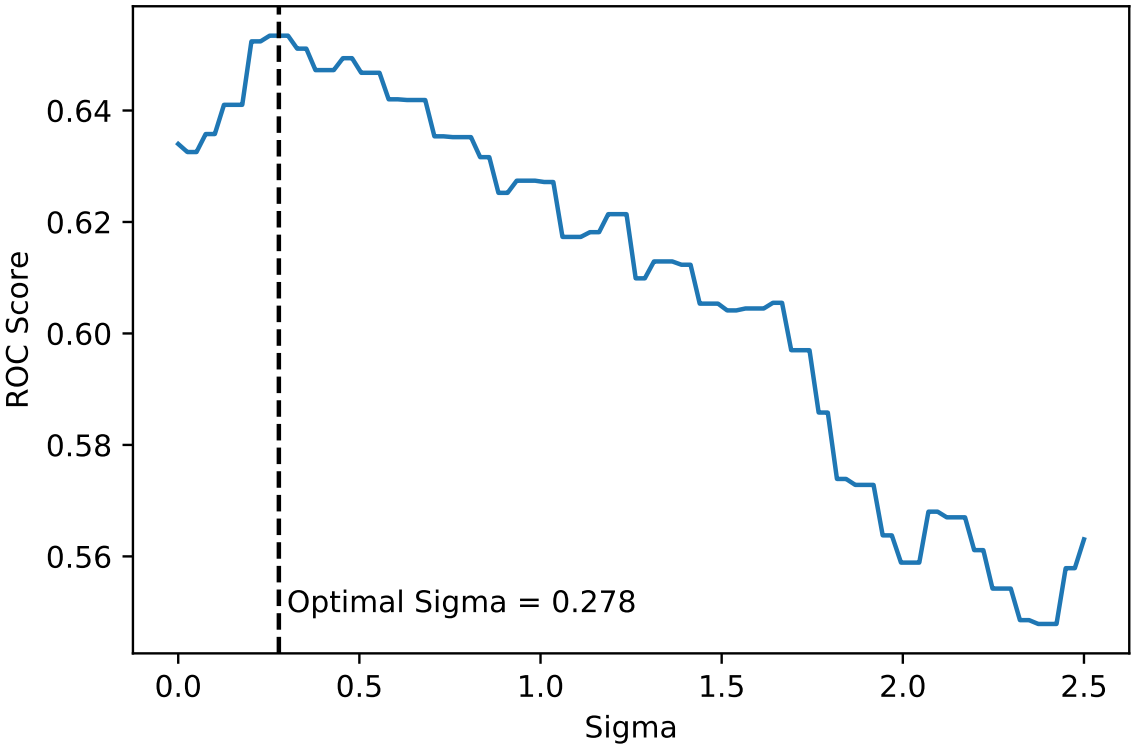


Figure 1.6: Naive Bayes Mutinomial classifier

Chapter 2

Natural language processing

2.1 Text normalization

The first stage in the implementation is to perform the normalization of the data, in this case the text.

Numbers are substituted by a flag term. In the data numbers occur when the transcript is unavailable due to unintelligibility.

All words are converted to lowercase in order to homogenize.

2.2 Term Frequency - Inverse Document Frequency

The text data is treated as a Bag of Words. Each document is encoded in a matrix where each column corresponds to one of the words that it contains. After this is done for all documents, the count is divided by the number of occurrences of that word in all documents, so as to obtain a relative importance of each word. Thus the words that are common will have a lower weighting than words that are rare.

By doing this we are assuming that the rare words are more significant than the others in the classification algorithm. Under this assumption, this is a feature extraction process.

After several tests, the results were not significantly improved by using n-grams larger than 1. This makes the matrix around 13,000 columns wide, which correspond to each one of the different words.

2.3 Model selection

In order to find the appropriate model for the classifier the most obvious was using a Support Vector Classifier. Support Vector Machines in general perform very well in high.

As a baseline to compare with, a multinomial Naive Bayes classifier was used. This was chosen because of its simplicity, especially in the Grid Search algorithm, as it has only one intrinsic parameter to optimize.

Aside from this, as is the case with high dimensionality a Generalized Linear Model with Ridge Regression was used [4].

Finally, some test were conducted with a Random Forest classifier in order to test out the computation costs of the Gris Search. A Random Forest contains many parameters that can be tuned, e.g. the number of trees, the depth of the trees, the threshold for the number of samples required to split a node or a leaf and bootstrapping.

Of course, this increases the computation cost exponentially.

2.4 Scoring techniques

In assessing the performance of the models, several scoring techniques were used.

- **Area Under the Receiver Operating Characteristic Curve** obtained by plotting the True Positive Rate against the False Positive Rate. The output value is the area under the curve.
- **Accuracy** represents the correct labeling, calculated as $1/(\text{Total population}) \cdot (\text{true positive} + \text{true negative})$.
- **Precision** or Positive Predictive Value are the correct positive predictions related to all positive predictions, calculated as

$$P = (\text{true positive})/(\text{true positive} + \text{false positive})$$

- **Recall** are the correct instances of all positive predictions, calculated as

$$R = (\text{true positive})/(\text{true positive} + \text{false negative})$$

- **F1 score** as the harmonic mean of Precision and Recall

$$F1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

As well as these indicators, Confusion Matrices have also been used in order to assess if the class imbalances were causing all predictions to be single class during the initial test stages.

Chapter 3

Experimental methodology

3.1 Crossvalidation

The results so far are found relying on the data from the given train/test split (4000/2000). K-fold cross-validation on the training and testing sets is performed in order to provide a more solid result and to avoid overfitting. The number of folds will depend inversely on the number of parameters in the model.

In this case a 3-fold crossvalidation is conducted. This relatively low number was decided based on the requirement of having large enough data to highlight the relevant parameters in each fold.

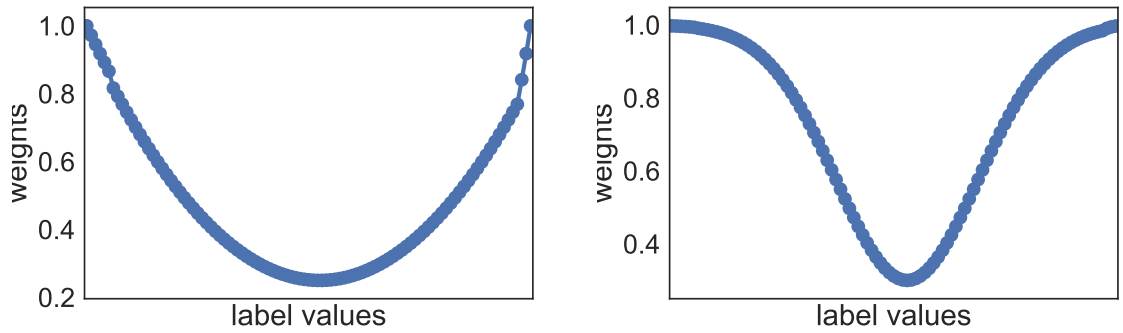
In order for the data to remain balanced, the folds are stratified keeping the same proportion of labels than in the original data. This is done to reduce the negative effect of class imbalance. However, we are assuming that the data is independent identically distributed.

The resulting model will be an average of each one of the models generated in each fold.

3.2 Parameter Grid-Search

A fine tuning of the algorithm's parameters is required, as an implementation of the study done in the previous sections. A Grid-Search algorithm is implemented to establish the values of the parameters of each tested model. For example, the γ and C for the Support Vector Classifier, α in the case of a Naive Bayes multinomial classifier, or n estimators, max features, max depth, min samples split, min samples leaf and bootstrap in the case of a Random Forest classifier, which can be either numeric or logical.

Grid-Search cannot be performed normally on a cross-validation setting. If the parameters are tuned using Grid-Search and then the model is trained with the optimal given parameters, the accuracy results would be artificially inflated. That is why the parameters will be crossvalidated on the trainig set, evaluated using the test set. The model will be then trained again using the train and test sets using crossvalidation and scored using the previously unseen validation set for which the ground truth is also available.



(a) Calculated weights vs label values using a parabolic function, with the minimum weight optimized at 0.25.

(b) Calculated weights vs label values using the complementary to a Normal distribution, with the minimum weight optimized at 0.3.

Figure 3.1

3.3 Weighting the labels

As seen in the previous section, the documents with ambiguous labels are problematic for training, but removing them altogether poses other problems we have already seen and decreases the data size. To address this, an alternative method was devised.

The labels that are further from the mean value are expected to be more valuable for the classifier model because they are more likely to contain features with higher contrast. In order to feed this knowledge to the model, a weight is matched to each label proportional to its value. There are two functions that have been used; a parabolic function and a complementary Normal distribution.

The parabola is defined as having values $y = 1$ both when label values are $x = 0$ and $x = 1$, while having the vertex aligned with the median of the labels values x =median. The only free parameter of the parabola is the minimum weight, i.e., what will the y value be when x =median. This is used as a model hyper-parameter and is optimized together with the intrinsic parameters of each model.

$$y = ax^2 + bx + c$$

By imposing said conditions, $c = 1$, $b = -a$ and $a = \frac{w-1}{m^2-m}$, where w is the vertex of the parabola or minimum weight and m is the mean of the label values.

The complementary to a Normal distribution is calculated in a similar way. First, a standard normal distribution is fit to the label values as follows:

$$y = \frac{1}{\sqrt{2\pi s^2}} e^{-\frac{(x-m)^2}{2s^2}}$$

where m is again the mean and s is the standard deviation of the label values. Then, a normalization process is applied to the list of y values:

$$y_{\text{normalized}} = 1 - \frac{y}{\max(y)}(1 - w)$$

This scales the weight values inside the $[0, 1]$ interval and incorporates the w minimum weight, as in the parabola function, to be optimized as a hyper-parameter of the model. This factor is then subtracted from 1 in order to obtain the complimentary values.

In Table 4.2 and Table 4.3 the results for the weighting of the labels modifies the F score.

3.4 Principal Components Analysis

The tokenized TFIDF matrix of 1-grams is about 13,000 columns wide. This means that there are 13,000 different words that are used as features for classification (Note that this is without lematization).

In order to reduce dimensionality of this matrix, improve the computation time and extract more valuable features, a Principal Components Analysis method is applied to the matrix (as seen in [2]), which in the case of text can also be referred to Latent Semantic Analysis. As its name suggests, the intention behind this is highlighting the latent structure in the combinations of words that might determine the labeling accurately.

This is in fact a Truncated Singular Value Decomposition that effectively projects the large matrix into a smaller space of a given size. This performs a dimensionality reduction by means of Truncated Singular Value Decomposition. Contrary to PCA, this estimator does not center the data before computing the SVD, so it can work with sparse matrix data types more efficiently [4].

The size of the transformed space is arbitrary but of course can greatly affect the performance of the model later on. As reference, the space is reduced to a useful size when it is around the square root of the original. In this case, 13,000 columns will be projected onto just 114.

Chapter 4

Conclusion and further steps

4.1 Comparative Results

Data normalization Table 4.1,

- the highest performance is attained when keeping the stopwords
- the flagging of numbers and punctuation does not affect the result significantly, and in labels other than *interview* it further confuses the results.
- for larger n-gram combinations, the results are not improved significantly either.

Parabolic vs complimentary normal weights Table 4.2, 4.3.

- contrary to intuition, some of the values show inferior results as in the previous section. This is the case because the tuning of the parameters is evaluated with a different dataset than the final scoring dataset. This way the score not biased and is robust against new data.
- the results increase slightly using the complimentary normal vs the parabolic weighting
- ultimately the optimal weighting for each case doesn't significantly improve or worsen the results from a fit prior class imbalance in the Naive Bayes Multinomial model.
- Results after applying weights vary slightly depending on the used model. For example, in the case of the Ridge Regression model, parabolic weights allow for a higher performance than complimentary Normal weights.

	With Stopwords	Normalized and flagged text With stopwords	Normalized and flagged text Without stopwords
extraversion	0.611293	0.598147	0.580981
neuroticism	0.628616	0.622925	0.608092
agreeableness	0.625293	0.634446	0.6221
conscientiousness	0.578556	0.573936	0.550909
openness	0.596273	0.592779	0.591751
interview	0.625637	**0.628705**	0.600904

Table 4.1: F1 score effects with text normalization.

	Ridge Regression Parabolic Weights	Ridge Regression Complimentary Normal weights
extraversion	0.552336	0.541121
neuroticism	0.616121	0.553447
agreeableness	0.681471	0.559856
conscientiousness	0.603965	0.541099
openness	0.67349	0.543438
interview	0.571041	0.564878

Table 4.2: F1 score effects with weighting function using a Ridge Regression model.

	Parabolic weights	Complimentary Normal Weights	Fit prior class imabalance
extraversion	0.598147	0.588332	0.598147
neuroticism	0.620936	0.649611	0.622925
agreeableness	0.626318	0.636713	0.634446
conscientiousness	0.56634	0.564312	0.573936
openness	0.592593	0.600977	0.592779
interview	0.620192	0.62569	**0.628705**

Table 4.3: F1 score effects with weighting function using the Multinomial Naive Bayes model as in the other cases.

Improvement by using PCA Table 4.4.

- The Random Forest model obtains very weak results feeding it the TFIDF matrix directly but increases significantly by using PCA instead.
- The run time for the hyper-parameter optimization is a constraint for this model's use. Optimization should be done in a faster machine.

4.2 Discussion

1. Complimentary normal weights produce slightly improved results over the parabolic in the Naive Bayes Multinomial model. The reason for this is most likely to be the decreased imbalance in the complimentary Normal thanks to higher probability of large weights.
2. Stopwords provide a slightly higher F1 score. This might be due to 1, the tone of the language used, ranging from relaxed to colloquial, and 2, the classification is not aiming at a

	Random Forest with PCA	Random Forest without PCA
extraversion	0.540856	0.548604
neuroticism	0.600083	0.581646
agreeableness	0.540641	0.539638
conscientiousness	0.567616	0.519431
openness	0.618565	0.528736
interview	0.617455	0.538425

Table 4.4: F1 score using a Random Forest model, with and without PCA.

subject classification but something more abstract that is not necessarily identifiable in the vocabulary alone.

3. PCA improves results in a similar magnitude as weighting does in general, although the end performance is model specific. This suggests the existence of an upper bound for this dataset, in which both fail at extracting significant features from the ambiguous labels.
4. Class imbalance is not a significant problem.
5. The maximum performance is low because the majority of the labels are indeed close to the center in both the training and the validation. The main cause of low performance is the difficulty in useful parameter extraction due to the ambiguity of the ground truth.
6. Different models have different behaviors depending on the methods used.

4.3 Further steps

- Use flagging for more markers in the text, especially, punctuation.
- Investigate if semi supervised Latent Dirichlet Analysis in order to both optimize feature extraction and select which of the labels in the training set should be discarded increases the performance.
- Use κ agreement in order to discard labels. The expectation is that the disagreement would be very high, and thus the resulting training set would be much smaller, [3]
- Parallelize the minimum weight parameter Grid-Search to improve performance of the hyper-parameter optimization, especially in the Random Forest models.
- Use the prosodic and face recognition data provided in the full dataset.

List of Figures

List of Tables

Bibliography

[1] Tupes, E.C., Christal, R.E. *Recurrent Personality Factors Based on Trait Ratings*, Technical Report ASD-TR-61-97, Lackland Air Force Base, TX: Personnel Laboratory, Air Force Systems Command, 1961.

[2] Jordi Luque, Carlos Segura, Ariadna Sánchez, Martí Umbert, Luis Angel Galindo, *The role of linguistic and prosodic cues on the prediction of self-reported satisfaction in contact centre phone calls*, 2017.

[3] Víctor Ponce-López, Baiyu Chen, Marc Oliu, Ciprian Cornearu, Albert Clapés, Isabelle Guyon, Xavier Baró, Hugo Jair Escalante, Sergio Escalera *ChaLearn LAP 2016: First Round Challenge on First Impressions - Dataset and Results*, 2016.

[4] Pedregosa, F. Varoquaux, G. Gramfort, A. Michel, V. Thirion, B. Grisel, O. Blondel, M. Prettenhofer, P. Weiss, R. Dubourg, V. Vanderplas, J. Passos, A. Cournapeau, D. Brucher, M. Perrot, M. Duchesnay, E., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, volume 12, pages 2825-2830, 2011

[5] Fleiss, J. L. *Measuring nominal scale agreement among many raters*, Psychological Bulletin, Vol. 76, No. 5, pp. 378–382, 1971

[6] Hunter, David R. *MM algorithms for generalized Bradley–Terry models*, The Annals of Statistics, 32 (1): 384–406, doi:10.2307/3448514, JSTOR 3448514, 2004.