Assignment Data Mining 2025:
Classification for the Detection of Opinion Spam

Instructions

This assignment should be completed by teams of three students. Register your team in Brightspace as soon as possible. Carefully read the complete assignment, including appendices, before you start working on it.

1 Introduction

Consumers increasingly review and rate products online. Examples of review websites are TripAdvisor, Yelp and Rate Your Music. With the growing popularity of these review websites, there comes an increasing potential for monetary gain through opinion spam: fake positive reviews of a company's product, or fake negative reviews of a competitor's product. In this assignment, we address the problem of detecting such deceptive opinion spam. Can you tell the fake reviews that have been deliberately written to sound authentic from the genuine truthful reviews?

2 The Data

We analyze fake and genuine hotel reviews that have been collected by Myle Ott and others [1, 2]. The genuine reviews have been collected from several popular online review communities. The fake reviews have been obtained from Mechanical Turk. There are 400 reviews in each of the categories: positive truthful, positive deceptive, negative truthful, negative deceptive. We will focus on the negative reviews and try to discriminate between truthful and deceptive reviews. Hence, the total number of reviews in our dataset is 800. For further information, read the articles of Ott et al. [1, 2].

3 Analysis

Ott analyses the data with linear classifiers (naive Bayes and Support Vector Machines with linear kernel). Perhaps the predictive performance can be improved by training a more flexible classifier. We will analyse the data with:

1. Multinomial naive Bayes (generative linear classifier),
2. Logistic regression with Lasso penalty (discriminative linear classifier),
3. Single classification trees, (non-linear classifier) and

4. Random forests (ensemble of trees).
5. Gradient boosting (ensemble of trees).

We use folds 1-4 (640 reviews) for training and hyper-parameter tuning. Fold 5 (160 reviews) is used to estimate the performance of the classifiers that were selected on the training set. Use cross-validation or (for random forests) out-of-bag evaluation to select the values of the hyperparameters of the algorithms on the training set. You are not required to use the original 4 folds in cross-validation, you may for example create 10 new folds. For naive Bayes, the performance might be improved by applying some form of feature selection (in addition to removing the sparse terms). The other algorithms (trees, regularized logistic regression) have feature selection already built-in, but might still benefit from removal of sparse terms. Examples of hyperparameters are:

- λ (or $C = \frac{1}{\lambda}$) for regularized logistic regression,
- the cost-complexity pruning parameter α (called CP in `rpart`) for classification trees,
- the number of trees, and the number of randomly selected features for random forests,
- the shrinkage parameter (or learning rate) and tree depth in boosting,
- the value of k in selecting the top-k features for multinomial naive Bayes.

With hyperparameters we mean parameters that influence the complexity of the model, not just any parameter of the `python` or `R` function that you call to train the model. For example, the optimization algorithm used to fit a logistic regression model is not a hyperparameter.

You will have to make a number of choices concerning text pre-processing, feature selection, etc. You are not required to try all alternatives, but it is important that you clearly describe (and if at all possible, motivate) the choices you have made, so an interested reader would be able to reproduce your analysis. To measure the performance, use accuracy, precision, recall and the F_1 score.

You should address the following questions:

1. How does the performance of the generative linear model (multinomial naive Bayes) compare to the discriminative linear model (regularized logistic regression)?
2. Are random forests and gradient boosting able to improve on the performance of the linear classifiers?
3. Does performance improve by adding bigram features, instead of using just unigrams?
4. What are the five most important terms (features) pointing towards a fake review, and what are the five most important terms pointing towards a genuine review?

All in all, the test set should only be used to estimate the performance of ten models: the selected multinomial naive Bayes, logistic regression, classification tree, random forest, and gradient boosting models, with and without bigram features added. Comparisons of the performance of different models should be supported by appropriate statistical significance tests.

4 Data Analysis Software

You are allowed to use `Python` or `R` to perform the analysis for this assignment.

4.1 Using Python

The library `scikit-learn` contains some useful functions for feature extraction from text (see: `https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction`) It also contains implementations of many machine learning algorithms. For example, you can use `sklearn.linear_model.LogisticRegression` for regularized logistic regression. The penalty should be set to 'l1' to get Lasso regularization, but note that the regularization parameter $C = \frac{1}{\lambda}$. For tuning of the regularization parameter C, you can consider using `sklearn.linear_model.LogisticRegressionCV`.
The Multinomial Naive Bayes model is implemented in `sklearn.naive_bayes.MultinomialNB`. Gradient boosting can be performed with `sklearn.ensemble.GradientBoostingClassifier`. Python also has several specialized libraries for natural language processing, such as `NLTK`.

4.2 Using R

We recommend the `tm` package for pre-processing the text corpus, and creating a document-term matrix. For regularized logistic regression we recommend the package `glmnet`, in particular its function `cv.glmnet` for tuning hyperparameters through cross-validation. For multinomial naive Bayes, you can use the function `multinomial_naive_bayes` from the `naivebayes` package. For classification trees you can use `rpart` and for random forests the `randomForest` package. For gradient boosting, the package `xgboost` is recommended. Read the documentation of the packages to learn about their possibilities.

5 The Report

The report should be written as a paper reporting on an empirical data mining study. This means there should be a proper introduction motivating the problem, a section describing the data that was used in the study, a section describing the setup of the experiments and their results, and a section in which the conclusions are presented. The experiments should be described in sufficient detail, so that the interested reader would be able to reproduce your analysis. For examples, see the papers of Ott et al. [1, 2]. There is no page limit.

Please answer the following question at the end of your report (5-10 sentences is enough): Did you use any generative AI tools (e.g., ChatGPT, Copilot, Gemini) while working on this assignment? If yes, describe how you used them and what changes or decisions you made based on that input. (e.g., I asked ChatGPT how best to measure feature importance in random forests, and I followed its advice to use method ...). If no, simply state that you did not use them. This reflection will not affect your grade directly, but failure to disclose AI use may.

References

[1] Myle Ott, Yejin Choi, Claire Cardie and Jeffrey T. Hancock, *Finding deceptive opinion spam by any stretch of the imagination.* Proceedings of the 49th meeting of the association for computational linguistics, pp. 309-319, 2011.

[2] Myle Ott, Claire Cardie and Jeffrey T. Hancock, *Negative deceptive opinion spam.* Proceedings of NAACL-HLT 2013, pp. 497-501, 2013.

Appendix

A Note on Using Generative AI Tools

You may use generative AI tools (e.g., ChatGPT, Copilot, Gemini) as learning aids, but not as a substitute for your own work. You are allowed to ask for explanations of concepts (e.g., TF-IDF, lemmatization), seeking tips for appropriate statistical tests, brainstorming analysis strategies, or polishing the writing of your report. It is not allowed to upload the dataset for a full solution, to submit AI-generated code or analysis as your own, or to let AI write your final report.

You must understand and be able to explain every part of your analysis. If you cannot explain your code, methods, or results, you will not receive full credit. Your grade will depend on the quality of your reasoning, analysis, and explanation, not just model performance.

B Code Submission Instructions

Along with your written report, please submit the code you used to run the experiments. Use the following format:

1. Scripts
   - (a) Submit your code as one or more `.py` (Python) or `.R` (R) files.
   - (b) Organize them clearly (e.g., `preprocessing.py, train_model.py`).
2. Include a `README.md` file that explains:
   - (a) How to run your code (e.g., `python main.py`).
   - (b) Which script(s) produce the key results in your report (e.g., The performance measures reported in Table 2 are produced by `train_model.py`).
   - (c) Which external libraries/packages are needed to run your code.
3. Reproducibility
   - (a) Your code should run start-to-finish on the provided dataset and reproduce the results you describe in your report.
   - (b) Make sure your submission is clean (remove unused code, temporary files, etc.).

C Handing in your work

Summarizing, the following items should be handed in:

1. A `.pdf` file with the report.
2. One or more `.py` or `.R` files with the code to run the experiments that you performed.
3. A `README.md` file explaining how to run your code.

These files should be collected in a `.zip` file, which should be handed in through Brightspace.