# DAT340 - Applied Machine Learning
# Programming Assignment 3C

**Stefano Ribes**
Chalmers University of Technology
`ribes@chalmers.se`

## 1 Introduction

This report describes the work done for the third programming assignment in the course *DAT340 Applied Machine Learning*. The main objective of the assignment was to study, design and evaluate a machine learning system capable of annotating crowd-sourced restaurant reviews. In particular, the model should be able to classify a given input review as positive or negative.

In order to design such system, I performed an initial cleaning processing of the provided data. Next, I designed and cross-validated a series of data pipelines, each including a word vectorizer and a classification model. The analysis focused on rather simple models, such as: linear models, decision trees and ensembles. In the end, I selected the best models evaluated them against an accuracy metric. The best model was able to achieve 95% accuracy on the test dataset.

The remainder of this document is organized as follows: Section 2 illustrates the followed methodology in designing the machine learning system; Section 3 provides instead a detailed evaluation of the system and shows the results of the conducted experiments. Finally, Section 4 gives an overview of the conclusions.

## 2 Methods

This Section summarizes the followed design methods. In particular, it describes the data preprocessing, the word vectorizer parametrization and finally the selected classification models and their configuration.

### 2.1 Data Cleaning

The provided data were collected and annotated by a large group of human workers and it was provided already split in a training and testing datasets. In particular, the training dataset presented a pair of annotations for each of the restaurants reviews. The reviews are set to become the system inputs, whereas the cleaned annotations will be used as labels.

The first step in the cleaning process was to analyze the two annotation lists and resolve the conflicting ones. The number of conflicting annotations accounted for 5.6% of the total number of training reviews, meaning that there was already a very high consensus between annotators.

In order reduce the conflicting annotations, I started from setting all annotations to either 0 or 1, since I noticed that there were annotations labeled as -1, 4, 9, and other values. I assumed negative values as negative reviews and marked them as 0, whilst any other positive number was capped to 1. Having done that, the number of discording annotations lowered to 4.2%, *i.e.* 295 annotation pairs. A possible way to further resolve such conflicts would be to manually choose a true label for such annotations or rely on accurate classifiers.

Due to a lack of time, and since the amount of conflicts was relatively low, I decided to simply drop the both the conflicting pairs and their corresponding reviews from the dataset.

### 2.2 Data Pipeline

In order to facilitate the training and evaluation of the models, I decided to encapsulate a word vectorizer and a classifier into a Sklearn data pipeline.

#### 2.2.1 Word Vectorizer

The word vectorizer (also simply referred as Vectorizer in the remainder of the report) is responsible for generating the model features. Text data, in particular, presents a varying input size, which makes it unpractical to feed to a classification model. Because of that, a common strategy is to construct a vector of numerical features which can then be trained on the available set of documents[1]. The final trained features will represent the count

[1] In our case, a 'document' is a single review.

of each word (or bigram) in the training documents and can eventually be normalized by the occurrence frequency of such words.

For my experiments, I utilized a count vectorizer followed by a Tf-idf normalization, both wrapped into a single Sklean `TfidfVectorizer()` class.

#### 2.2.2 Models

In this work, I experimented with the following set of classifiers:

- Linear models: Perceptron (baseline), Logistic Regression, SGD classifier, linear SVC, Ridge
- Naïve Bayes: BernoulliNB
- Trees (and Ensembles): Decision Tree Classifier, Random Forest
- Neural network: Multi-Layer Perceptron (MLP)

## 3 Evaluation

This Section reports the evaluation and experimental results of the designed and implemented system.

### 3.1 Cross-Validation

In order to compare the performance of the classifiers and select the best one, I exploited a *grid search* Cross-Validation (CV) strategy. The grid search allowed me to perform hyperparameter tuning according to a set of predefined configuration settings. In order to run a grid search I initially setup a common parameter grid for the Vectorizer. In particular, I limited the exploration to uni- and bi-grams, whether to use *idf* normalization, and limited the number of maximum features. Then, I added the parameters configurations for each of the classifiers. The parameters to tune are tailored to the type of classifier and some variables were to be chosen from a pool of definite numerical values[2]. The cross-validated models were scored according to accuracy and ROC-AUC, whereas the best estimator per grid search was selected based on its accuracy score. During CV, the data was split in 5 folds.

[2]For a future more in depth analysis, Sklearn provides a way to sample such numerical values from a specified distribution.

Table 1: Vectorizer configurations.

| Classifier | idf | n-grams | Max Features | Eval Acc. |
|---|---|---|---|---|
| Perceptron | False | 1, 2 | 5000 | 0.906 |
| BernoulliNB | True | 1, 2 | 5000 | 0.924 |
| DecisionTree | True | 1 | 1000 | 0.848 |
| RandomForest | False | 1, 2 | All | 0.934 |
| LogisticRegr. | True | 1 | All | 0.951 |
| SGD | True | 1, 2 | All | 0.949 |
| LinearSVC | True | 1, 2 | All | 0.950 |
| Ridge | True | 1, 2 | All | 0.949 |
| MLP | False | 1, 2 | All | 0.953 |

### 3.2 Accuracy

The cross-validation accuracy scores of the best classifiers are showed in Figure 1. The perceptron, acting as baseline, performs comparatively high at 91.1%, closely followed by the BernoulliNB and Random Forest models at 92.5% and 93.5% respectively. The Decision Tree shows the worst score at 87.1%. On the other hand, the rest of the classifiers achieved an accuracy of around 95%, with the Ridge classifier being the best at 95.7%. For more detailed information, please refer to the attached Jupyter Notebook.

Figure 2 illustrates instead a comparison of the classifiers against the test dataset in terms of accuracy. The detailed performance scores are reported in Table 1. Despite the Ridge classifier being the best in CV, the best classifier on the test dataset is instead the MLP classifier with 95.3% score. What's interesting to notice in Table 1, is the type of the Vectorizer which provided the best results. It seems that considering both uni- and bigrams and applying idf weighting might improve the accuracy score of the classifiers. More noticeable, the best performing classifiers considered all the generated features, even though this might be due to the limited exploration range of the configuration (being 1000, 5000 or all, around 100k).

Table 2 shows the predictions of the classifiers when evaluating the review "not recommended". The review is mislabeled as a good comment despite being a negative one. I believe that the word "recommended" fools some of the models into believing it's a positive review, since it's typically associated to good restaurants.

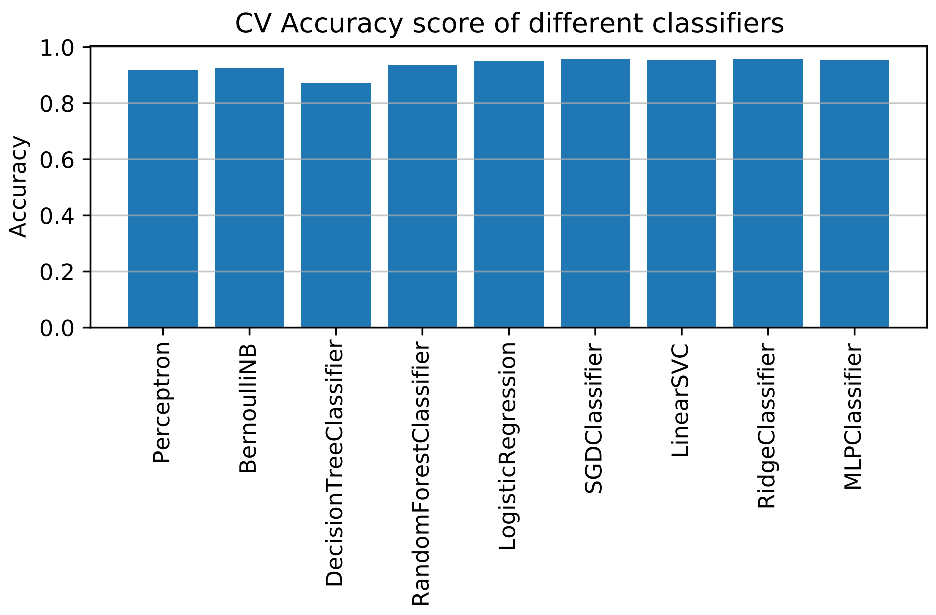

Figure 1: CV accuracy of the analyzed classifiers.

Table 2: Predictions of a challenging review.

| **Classifier** | **True Label** | **Predicted Label** | **Confidence** | **Review** |
|---|---|---|---|---|
| Perceptron | 0 | 1 | N.A. | not recommended |
| BernoulliNB | 0 | 1 | 0.992279 | not recommended |
| Decision Tree | 0 | 0 | 0.924347 | not recommended |
| Random Forest | 0 | 0 | 0.532642 | not recommended |
| LogisticRegr. | 0 | 1 | 0.710735 | not recommended |
| SGDClassifier | 0 | 1 | N.A. | not recommended |
| LinearSVC | 0 | 1 | N.A. | not recommended |
| Ridge | 0 | 1 | N.A. | not recommended |
| MLP | 0 | 0 | 0.965501 | not recommended |

### 3.3 Feature Importance

By analyzing the feature importance of the Random Forest classifier, the top 5 features we find 1-gram adjectives: ”great”, ”delicious”, ”amazing”, ”excellent”, ”worst”. I expected negated verbs like ”don”, ”wasn” and ”didn” to be ranked higher, but they are still in the top-50 positions. At the bottom there are very ”strong” bigrams such as ”disgusting behaviour”, but apparently they are also very rare in the documents and so ranked low.

For the Decision Tree classifier instead, despite exploiting lesser number of features (1000 versus 10000), the most important features remain the positive adjectives. At the bottom are lesser important features like food names like ”gyoza”, which is a reasonable and intuitive assumption.

## 4 Conclusions

This reported described the design, implementation and evaluation of an AI system able to classify restaurant reviews. The system was implemented by utilizing a word Vectorizer and a classification model. In terms of accuracy, the best performing model was a MLP classifier which achieved a test score of 95.3%.

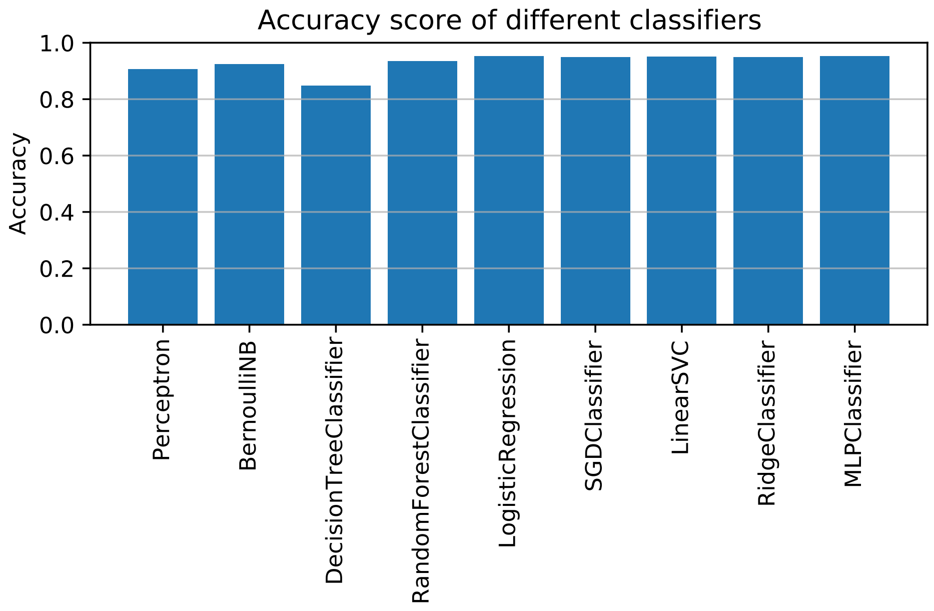

Figure 2: Test accuracy of the analyzed classifiers.