Katholieke
Universiteit
Leuven

**Faculty of Engineering Science**
**Master of Artificial Intelligence**

# BIG DATA ANALYTICS PROGRAMMING

Assignment 1: Online Spam Filters

**Papadopoulou Maria**
r0977221
maria.papadopoulou@student.kuleuven.be

## Contents

# 1 Spam Filtering

## 1.1 Evaluation Metrics

Spam Filtering is a simple classification problem. In a classification problem, the most common metrics to measure efficacy are accuracy, recall, precision and F1-score.

$$Accuracy = \frac{\text{true positives} + \text{true negatives}}{\text{total classified}}$$

Accuracy measures the overall correctness of the model. However it does not seem an appropriate evaluation metric for imbalanced classification problems like spam filtering, where the number of spam classified emails is very small in comparison to ham classified emails.

$$Recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Recall concentrates the metric in one class only and measures how well the system predicts it. In our specific situation, we focus on the spam emails and measure the number of correctly classified spams out of all spams.

$$Precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Precision represents how precise the system is when annotating an email as "spam". In our specific situation is the ratio of correctly classified spam emails to the number of emails classified as "spams". Precision is very important for our experiments because when the precision is high, close to 1, that means that the system classify emails as spams without errors.

$$F1 - score = 2 * \frac{\text{precision * recall}}{\text{precision} + \text{recall}}$$

F1-score is the harmonic mean of precision and recall. It is a metric of overall performance and very significant when there is an imbalance between the classes. High score implies that we have a good classifier where both recall and precision are high.

## 1.2 Experiments

Our experimental setup aims to identify the parameters that impact the performance of our models and understand how these parameters influence the outcomes. To do so, we run each algorithm with the default values to establish a baseline performance and then we started experimenting with different parameter values for finetuning. For the experiments and the results provided below, we set as threshold -1 for Naive Bayes with feature hashing, -4 for Naive Bayes with Count Min Sketch and 0 for both Perceptron implementations.

Significant note: For the analysis below,the notation 9,12,14 number of buckets was used which is actually implying $2^9$,$2^{12}$,$2^{14}$ number of buckets.

### 1.2.1 Naive Bayes with Feature Hashing

- Increasing ngrams decreases evaluation metrics
  The use of larger n-grams can lead to a higher risk of false negatives, because specific patterns or combinations of words represented by larger n-grams may not occur frequently enough. As a result, the model may struggle to generalize to instances that contain these less common patterns, leading to false negatives. False negatives impact recall, as we can see in Figure 1 recall decreases for larger n-grams.
- Adding more buckets increases evaluation metrics
  In Figure 2 we can see that for larger number of buckets, our model performs better. With more buckets the hash space becomes bigger, reducing the chance that different features will hash to the same buckets, reducing collisions and influence positively the overall performance.

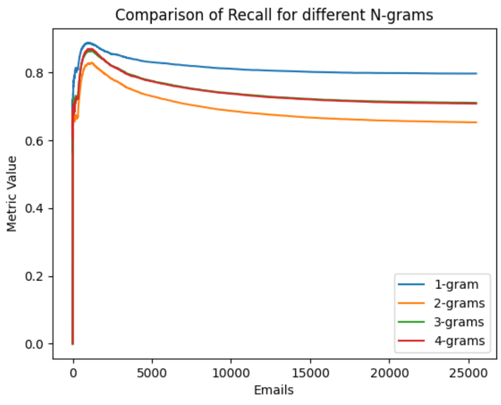


Figure 1

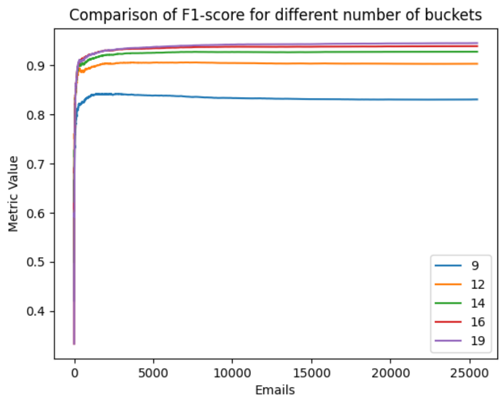
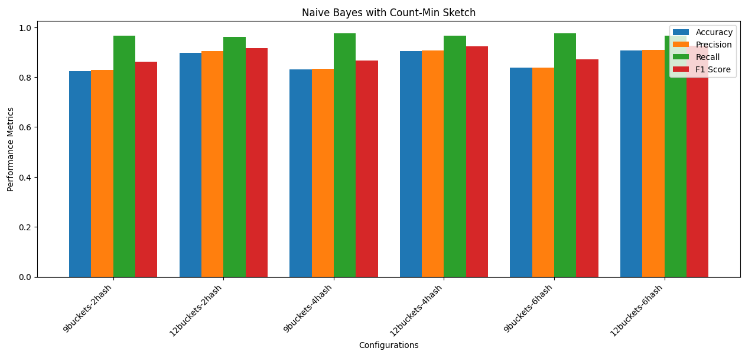


Figure 2

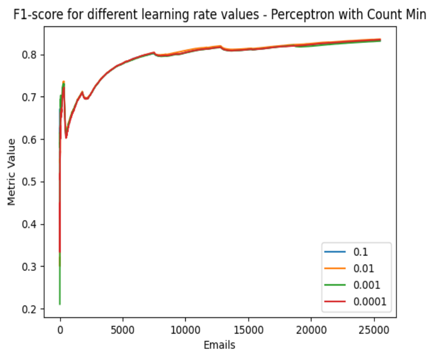
#### 1.2.2 Naive Bayes with Count-Min Sketch

- Adding hash functions
  We notice that adding more hash functions doesn't have significant changes to the evaluation metrics of our model. We conclude that adding more hash functions can improve discrimination between features, reducing the likelihood of collisions, especially for larger number of buckets. However, beyond a certain point, the additional benefit gained by adding more hash functions seems to become marginal.



#### 1.2.3 Perceptron Implementations

- Adjusting the learning rate
  Both Perceptron models don't seem to be highly sensitive to changes in the learning rate, which is important as we achieve convergence, as we can see in the figure below. We found are best results for learning rate 0.0001 for Perceptron with Count Min Sketch and 0.01 for Perceptron with Feature Hashing.



### 1.2.4 Comparison Experiments

In order to check the generality of our implementations and find an optimal configuration we run them under the same hyperparameters as seen in Table 1. As our data have quite significant imbalance, with spam class be more prevalent than ham class, we use F1-score to get a clearer picture of our results. We observed that all performed better for 1-grams with Naive Bayes using Count Min achieving the highest f1-score 93.6 %. We also observed that we achieved better results for Perceptron with Count Min Sketch for smaller number of buckets.

| Number of buckets | Number of hashes | NBFH | NBCM | PFH | PCM |
|---|---|---|---|---|---|
| 9 | 1 | 0.842 | | 0.807 | |
| 9 | 2 | | 0.862 | | 0.834 |
| 9 | 4 | | 0.868 | | 0.85 |
| 12 | 1 | 0.906 | | 0.821 | |
| 12 | 2 | | 0.917 | | 0.797 |
| 12 | 4 | | 0.923 | | 0.828 |
| 14 | 1 | 0.928 | | 0.82 | |
| 14 | 2 | | 0.933 | | 0.819 |
| 14 | 4 | | 0.936 | | 0.826 |

Table 1: Evaluation Metrics

Figure 5 and 6 show the comparison between spam filter implementations with Feature Hashing and Count Min Sketch. From the experiments, we conclude that both implementations with Count Min Sketch perform almost the same or even better than Feature hashing (especially for smaller number of buckets) as they are more memory efficient, minimizing the space requirements. For example, we can see that the Perceptron with Count Min achieves better results using only 9 buckets than Feature Hashing with 16 buckets.

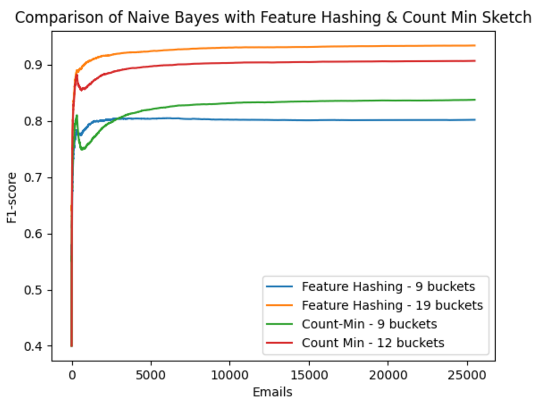


Figure 5

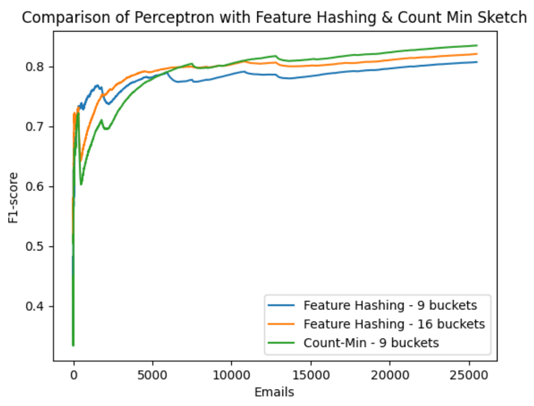


Figure 6

In coclusion, Naive Bayes implementations seem to have better overall performance than Perceptron.

## 2 Resources

https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/