Assignment 1

Arlind Iseni
aris20@student.bth.se
Alexander Jamal
aljm19@student.bth.se

I. Introduction

A commonality between email applications is their ability to detect spam email that are in the inbox. In this assignment we try to achieve the ability to detect whether an email is spam-or-ham by implementing a concept learner, in our case the least general generalization, a logical expression of the most general concepts between entities. We were given a data set of aggregated statistical summaries for each e-mail in a big cohort of e-mails [1]. The data type for most of these features were of the continuous type and an appropriate data-preprocessing with discretization as the main operation was to be conducted.

II. Method

A. Data Exploration

We examined the distribution of each feature. The plots followed a right-skewed distribution for almost all attributes, except for our target column *is_spam* which had a 40/60-procent ratio between spam and ham.

B. Data Cleaning

The data set had no missing values and no erronous values . However we identified 391 duplicate instances.

C. Data transformation (Discretization and space-complexity)

The concept learning algorithm requires discrete values for training, however most of our features were statistics with values between 0 and 1. We discretized our data into 10 bins with the uniform binning technique, leveraging the KBinsDiscretizer method from Sklearn [2]. The data was encoded ordinally so that the bins would be ordered according to value. We decided on these parameters by running an experiment on different binning techniques and bin amounts using accuracy (% of predictions that are correct) as our performance metric. As our values are in the range of 0-10, we decided to change the data type of all attributes to 8-byte integers to reduce space complexity.

D. Computing instance space, hypothesis space, and conjunctive space

Instance space: All possible or describable instances, whether they are present in our data set or not. To compute the instance space, we count the amount of bins each feature has and then multiply these values together to obtain the instance space [3].
- **Hypothesis space:** The possible permutations of a class. To calculate the hypothesis space, we take the number of classes to the power of the instance space.
- **Conjunctive space:** the number of possible conjunctive concepts [2]. We calculate the conjunctive space by taking the amount of bins per feature plus one and compute the multiplication of all these numbers [3].

E. Code and algorithm implementation

All code was written in python 3.10 in Jupyter-lab. The algorithm implementation was based on the pseudo-code of algorithm 4.1 and 4.2 from the course literature [3].

F. Training and evaluation (training, testing, accuracy, sensitivity, specificity)

We partition our data in an 80/20 split into two subsets, training and testing. After training we receive the conjunctive rule for our data set. This rule is a logical statement including complementary columns and their assigned values. We use this rule for prediction by subsetting our test data set according to our conjunctive rule and assigning these rows the predicted value of 1 (true). We evaluate our model's predictive performance by calculating the accuracy, sensitivity, and specificity.

III. Results

Our data transformation resulted in no different data distribution. Most of the features followed the same distribution as before. Furthermore, we decided to keep all the identified duplicates.

The *instance space* for our data set of 57 features and varying bins (1-10) is the following:

$$I=135966586618118146452252707632416030720000000000000$$

while the *hypothesis space*,

$$H=2^I$$

and finally, the *conjunctive space*,

$$C=1424507950081795870969288997392613376000000000000000000$$

The conjunctve rule we got is:

word_freq_hp = 0 ∧ word_freq_hpl = 0 ∧
word_freq_george = 0 ∧ word_freq_lab = 0 ∧
word_freq_857 = 0 ∧ word_freq_85 = 0 ∧
word_freq_cs = 0 ∧ word_freq_meeting = 0 ∧
word_freq_project = 0 ∧ word_freq_conference = 0.

From our analysis and implementation, we find an accuracy of roughly 60% with 10 bins and the uniform binning algorithm. All other binning algorithms (k-means, quantiles) and bin amounts (ranging from 2 to 20) resulted in lower accuracy.

IV. Conclusion

We can conclude that a concept learner-algorithm gives us a semi-reliable (> 50%) estimate of whether an e-mail is spam or not, however it is a very inaccurate tool and with some configurations a coin flip would have given us a more robust prediction.

References

[1] *UCI Machine Learning Repository: Spambase Data Set.* [Online]. https://archive.ics.uci.edu/ml/datasets/spambase

[2] J. Brownlee, "How to Use Discretization Transforms for Machine Learning," *MachineLearningMastery.com*, May 21, 2020. https://machinelearningmastery.com/discretization-transforms-for- (accessed Nov. 26, 2022).

[3] P. A. Flach, "Chapter 4," in *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*, Cambridge University Press, 2012

[4] "Classifying spam / ham - Naive Bayesian Classifier," *kaggle.com*. https://www.kaggle.com/code/aishwarya0206/classifying-spam-ham-naive-bayesian-classifier#Splitting-Train-&-Test-set-of-Train-data (accessed Dec. 15, 2022).