Design, Implementation and Evaluation of a Machine Learning Model for Spam Classification

TECHNICAL REPORT

Developed by:

Andreas Patsalos

U224N3172

University of Nicosia

For:

COMP244 – Machine Learning and Data Mining I

May 2023

Background:

The objective of the project was to design, implement, evaluate and optimize a machine learning model to be applied for email spam classification. Python was chosen for the design environment, via VSCode IDE.

The training data set that was given was a raw data set with two attributes, one being the content of the email and the other being the label. It consisted of 7514 entries of different emails, each with its own set of characteristics, some of them missing some values, missing the label, foreign language etc.

Experimentation had to be done to select the appropriate classification model. The chosen model should have had the highest accuracy in combination with being the most efficient timewise.

After the selection of the most appropriate model, the said model would then be re-evaluated so as to not split the given data set but rather train on the whole of it, then take an unknown, unlabelled data set and test its accuracy on that specific set.

Experimentation:

For experimenting I chose three classification models which are the Decision Tree, Support Vector Machine and the Naïve Bayes classifier. Each model has been implemented on its own.

The first step was the pre-processing step. The dataset was cleaned from punctuation, converted to lower case and got rid of unnecessary symbols. This step ensured that there would be no unwanted interference during the classification because what the model takes into consideration is the actual words in the attribute column, in this specific case. This was achieved with Python’s Pandas data analysis library and the String library. Two different functions of said library were called. Translate, to remove punctuation, Lower, to convert the important attribute column into lower case and finally String Replace from the String library, to replace unwanted symbols with whitespace.

Then follows a display of the first ten email samples so as to make sure that the data file was read correctly.

After that, text vectorization followed. The data set was converted into a bag-of-words model in order for the classifier model to process it. This was achieved with Scikit-Learn’s CountVectorizer function.

The chosen training method was the train-test split, specifically 80% of the entries were reserved for training and 20% for testing. In order to have the same train-test split every time the classifier ran, a fixed random state was set. This was achieved for all three models with the implementation of Scikit-Learn’s train test-split function.

Experimentation continued:

Next, after cleaning the data set and splitting the data into train and test sets, was the implementation of the classifier model. Each model was implemented from Scikit-Learn's classifier model library.

After predicting the label attribute of the test data set split, the results were exported on a new CSV file that distinguishes the actual label from the predicted label.

At the end of the implementation process, the accuracy of each model resulted as followed:

- Decision Tree classifier model: 94.4%
- Support Vector Machine classifier model: 94.2%
- Naïve Bayes classifier model: 98%

Taking this into consideration, the Naïve Bayes classifier model was chosen as the one to feed the unknown, unlabelled data set later on.

Alterations to the code were carried so as not to split the given data set into train and test but now train on the whole given data set and then be able to accept a new CSV file with unlabelled data and similar format and then predict on that. After that it should export a new CSV file displaying the actual label and the predicted label.

Observations

All three classifier models performed relatively closely in terms of both accuracy and efficiency (timewise).