**Intelligent Systems**

**The AI Application Project**

**Spam Filter**

* 1. **Related to evidence of understanding**
     1. **What is the functionality of your system?**

The project I wish to undertake would be a spam filter, this system will be a form of supervised learning which will predict or determine whether an incoming email or text is spam or not - to do this I will be using different classification algorithms that will classify emails to teach our model to detect spam or ham.

While researching different algorithms and classifiers I found the Naïve Baye, KNN (K Nearest Neighbor) and SVM (Support Vector Machine) to be the most interesting and were promoted as efficient spam filters so I decided to test which had the best accuracy, precision and recall.

* + 1. **What are the basic inputs and outputs?**

The input for our supervised machine learning algorithms would be the training data or a labelled dataset this will be used to predict or find the probability of falling into either our spam or ham categories/outputs.

**Data Preparation**

A screenshot of a computer

Description automatically generated with medium confidenceThe dataset I chose had 5573 rows meaning it had 5573 datapoints. Using 2 Pandas methods, .groupby() and .describe() we can get a count of all the data grouped by its classification - 4826 were non-spam(ham) and 747 were spam giving us roughly 14% spam emails.

A screenshot of a computer

Description automatically generated with medium confidenceThere were 415 duplicate datapoints - to clean the data we first check if we have any null fields in the spam data, we can use the .isnull() method followed by .sum() which will add them all up if there are any.

Removing duplicates was simple using Pandas .drop\_duplicates() method which will drop any duplicates from the data, the subset parameter requires it to check all columns and inplace will arrange the data structure once duplicates have been removed resulting in 5158 datapoints 4517 being ham and 641 being spam resulting in just under 13% spam. (pandas.pydata.org. 2022)

Chart, bar chart

Description automatically generatedUsing matplotlib and seaborn we can get a better visual representation of the spread.

(Seaborn Barplot 2020)

**Dataset used:** https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

* + 1. **What is the real-world problem that it solves?**

According to statista.com almost 50% of emails in 2021 were spam, filtering our emails whether at home or within an organization is essential in protecting ourselves from malicious actors and phishing attempts which also improves our quality of life while emailing and surfing.

Email and spam filters in general;

* Protect against malware/viruses.
* Protect against social engineering and phishing attacks.
* Save time on an individual basis and as a company – which could potentially improve workflow and communications.
* Protect your data and potentially customer/client other persons data.
* Generally customizable so you can apply your own filter with whitelisted and blacklisted addresses.

(Nines, F. 2018, Statista Research Department. 2022)

* 1. **Related to algorithm explanation**
     1. **Details regarding algorithm selection**

**Naïve Baye**

Naïve Bayes classifiers are a family of machine learning models/algorithms based on Bayes probability theorem, in the case of a spam filter it calculates the probability of a class (spam or ham) when given data (in this case emails or text).

For this project I used the multinomial Naïve Bayes algorithm which seemed the most appropriate for classification with word counts.

(Scikit-learn.org. 2019)

Naïve Baye assumes each word in an email or message is independent of each other and calculates the probability of each word being a part of spam or ham from previous events i.e. the training data set.

A picture containing text

Description automatically generated

(Gavrilova, Y. 2019)

The probability of A given B equals the probability of B given A is true multiplied by the probability of A divided by the probability of B – essentially the naïve baye algorithm relies on a frequency table of words and how often they appear in spam or ham emails.

A simple illustration, everything must have a value as it can’t predict using a zero probability:

|  |  |  |
| --- | --- | --- |
| **Frequency Table** | **Spam** | **Ham** |
| Password | 4 | 1 |
| Football | 1 | 4 |
| Picnic | 1 | 5 |

P(SPAM|”Password”) = P(“Password”|SPAM) \* P(SPAM) / P(“Password”)

P(“Password”|SPAM) = 4/6 = 0.66\* (The occurrences of ‘Password’ in spam divided by total spam)

P(SPAM) = 6/16 = 0.375 (Total spam / total count)

P(“Password”) = 5/16 = 0.3125 (Total occurrences of “Password”/ total count)

P(SPAM|”Password”) = (0.66 \* 0.375) / 0.3125

P(SPAM|”Password”) = 0.792 or 79.2% a high chance that the word password will be a part of spam in this tiny dataset.

Diagram

Description automatically generated

(Wei, Q. 2018)

**KNN**

K-Nearest Neighbors is a supervised learning algorithm and is used for classification and regression.

It is known as a “lazy” algorithm rather than training it as you would with other supervised learning algorithms it stores the dataset and plots all the cases/datapoints and when a new data entry is entered its classification will be determined by the distance from other datapoints and their classifications so unlike the Naïve Baye algorithm which each case is independent, KNN depends on previous datapoints and their locations.

* Diagram

  Description automatically generatedInitialise by selecting a number for *‘K’* i.e. the *‘K’* closest datapoints – It is the main deciding factor in the KNN algorithm some interesting results when changing the *‘K’* value with best results at a *k-value* of 3.
* Calculate the distances between the new data and the dataset, the default method using sklearn.neighbors uses the Euclidean approach.
* Sort its neighbors in ascending order.
* The class with the greatest number of votes will be its classification.

(Christopher, A. 2021)

**SVM**

Support vector machine classifiers create a ‘hyperplane’ or boundary between two different classes of data or vectors which will be classified, vectors are numbers which represent coordinates like the KNN algorithm except rather than calculating its *‘K’* number of neighbors you create a hyperplane between the two classifications and new data entries will be classified by which side of the hyperplane they land.

Chart, scatter chart

Description automatically generated

(MonkeyLearn. no date)

SVM works by separating a dataset as best it can, the distance between the hyperplane and the closest vector of each classification is called the ‘margin’.

To deal with non-linear and joined planes SVM uses kernels to manipulate the datapoints and segregate them.

Diagram

Description automatically generated

(TEKMAN, M.E.H.M.E.T. 2020)

* 1. **Related to evaluation**
     1. **How have you measured that your system is successful?**

To measure the performance of my system I used the accuracy\_score function from the sklearn.metrics package, this calculates the accuracy of my training data and my test data to compare the predicted labels to actual labels.

First, we import accuracy\_score

Next, we categorize the data numerically.

Text

Description automatically generated

And after splitting the test and training data we can finally print the results.

Graphical user interface

Description automatically generated

For each of the algorithms I created a confusion matrix – this enabled me to see where errors were in my model and is used to define performance in a classification algorithm.

I created confusion matrix’s visually using seaborn.heatmap or alternatively you can use sklearn.metrics built in function confusion\_matrix which will print a 2d array instead of a heatmap.

It lists the True positives (TP), True negative (TN), False positive (FP) and False negatives (FN) – in this run the SVM algorithm correctly classified 1128 positive datapoints, correctly classified 133 negative datapoints, incorrectly classified 7 datapoints as positive when they were negative and incorrectly classified 22 datapoints as negative when, they were positive.

Text

Description automatically generatedAnother useful way of measuring performance was another built in sklearn.metrics function called classification\_report, this will give a detailed report of precision, recall and f-1 score

Accuracy measures how often the algorithm is correct which is calculated by Correct predictions/total predictions.

Precision is the percentage of true positives, calculated by True positive/(True positive+False positive).

Recall is the percentage of predicted positives from all positive cases calculated by True positive/(True positive+False negatives).

F-score is the mean of precision and recall.

In this run of the SVM classifier (one of its best runs) we can see the accuracy, precision and recall are nearly perfect with ham emails, but ran noticeably worse using spam emails with a recall of 0.85 giving it an f-score of 0.91. (Kanstrén, T. 2021)

And finally manual testing was implemented where I would enter in my own spam and ham emails and see if they registered correctly – 1 = spam and 0 = ham.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedSample tests:

* + 1. **Results**

I was surprised how well these algorithms worked, from my testing the KNN algorithm ran the worse on average with a large range of accuracy and precision ranging from 70-90 while the Naive Bayes and SVM algorithms consistently stayed above 85% with the Naïve Bayes running the best.

The recall for the KNN algorithm is abysmal as well meaning it’s not very good at predicting positives which in turn gives it a lower F-score and meaning the NB and SVM algorithms have a lot fewer false positives and negatives.

Chart, bar chart

Description automatically generated

(Average results after 5 runs using the average metrics from ham and spam)

In conclusion I believe the best algorithm to be the Naïve Bayes algorithm it seems to be extremely effective, it had consistently the best accuracy, precision, recall and f-score even over the SVM algorithm. It was also arguably the easiest to implement with the KNN being the most cumbersome as to have a good KNN algorithm you need to find the right K-value and the SVM algorithm needing linear kernels.

**References**

**Report**

Christopher, A. (2021) *K-Nearest Neighbor*, *Medium*. The Startup. Available at: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4.

Dataset used: https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

Gavrilova, Y. (2019) *Bayesian probability*, *Naive Bayes Algorithm for Beginners*. serokell.io. Available at: https://serokell.io/blog/naive-bayes-classifiers.

Kanstrén, T. (2021) *A look at precision, recall, and F1-score*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec#:~:text=F1%2DScore%20is%20a%20measure,than%20the%20traditional%20arithmetic%20mean.

*MonkeyLearn*. (no date) *Text classification using support vector machines (SVM)* Available at: https://monkeylearn.com/text-classification-support-vector-machines-svm/.

Nines, F. (2018) *Spam filtering: Why it's important and how it works*, *Five Nines Blog*. Available at: https://blog.fivenines.com/spam-filtering-why-its-important-and-how-it-works.

pandas.pydata.org. (2022). *pandas.DataFrame.drop\_duplicates — pandas 1.2.4 documentation*. [online] Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop\_duplicates.html.

Scikit-learn.org. (2019). *sklearn.naive\_bayes.MultinomialNB — scikit-learn 0.22 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.MultinomialNB.html.

*Seaborn Barplot* (2020) *seaborn barplot - Python Tutorial*. Available at: https://pythonbasics.org/seaborn-barplot/.

TEKMAN, M.E.H.M.E.T. (2020) *Text classification: SVM explained*, *Kaggle*. Kaggle. Available at: https://www.kaggle.com/code/mehmetlaudatekman/text-classification-svm-explained.

Published by Statista Research Department (2022) *Spam e-mail traffic share 2021*, *Statista*. Available at: https://www.statista.com/statistics/420391/spam-email-traffic-share/.

Wei, Q. (2018) “Understanding of the naive bayes classifier in spam filtering,” *AIP Conference Proceedings* [Preprint]. Available at: https://doi.org/10.1063/1.5038979.

Wingate, R. (2020) *Naive Bayes sms example*, *Ryan Wingate*. Available at: https://ryanwingate.com/intro-to-machine-learning/supervised/naive-bayes-sms-example/.

**Code**

*Creating a Spam Filter using Naive Bayes* (2022) *YouTube*. Nick Stugard YouTube Channel. Available at: https://www.youtube.com/watch?v=2sXAYoPIz3A.

Kharwal, A. (2021) *Email spam detection with machine learning: Aman Kharwal*, *thecleverprogrammer*. Available at: https://thecleverprogrammer.com/2020/05/17/email-spam-detection-with-machine-learning/.

TEKMAN, M.E.H.M.E.T. (2020) *Text classification: SVM explained*, *Kaggle*. Kaggle. Available at: https://www.kaggle.com/code/mehmetlaudatekman/text-classification-svm-explained.

Wingate, R. (2020) *Naive Bayes sms example*, *Ryan Wingate*. Available at: https://ryanwingate.com/intro-to-machine-learning/supervised/naive-bayes-sms-example/.

Yeafi, A. (2021) *Complete Guide on Classification Algorithms*, *Kaggle*. Kaggle. Available at: https://www.kaggle.com/code/ashfakyeafi/complete-guide-on-classification-algorithms#Support-Vector-Machine-Classifier.