42172 AT1: SMS Spam Prediction

PART: 3

AUTHOR: CYRUS KWAN
STUDENT ID: 25466929

DATE: 13 OCTOBER

SEMESTER: SPRING

YEAR: 2024

UNIVERSITY OF TECHNOLOGY SYDNEY

LINK TO SOLUTION NOTEBOOK:
[Google Colab]
[GitHub]

Master of AI

Faculty of
Engineering & IT

UTS CRICOS 00099F

Table of Contents

Summary or Executive Summary

Purpose
Compares the performance of modern classification models to those recommended in previous literature with the goal of classifying whether a given SMS text message is spam or non-spam.

Techniques
Analyses the classification task by using a support vector machine (SVM) classifier and a neural network and compares the model performance using precision, recall, f1-score, and accuracy.

Findings
This report found that neural networks outperform SVM classifiers in all metrics but precision where both models achieved a precision score of 1.00.

Conclusion
Spam SMS texts cause frustrations among users and unwanted load on IT infrastructure. Thus, it is necessary to filter spam messages to increase satisfaction for all parties. The results of this study contradict the recommendation in previous work that recommended an SVM classifier for spam filtering tasks

Recommendation
Users should use a simple neural network solution targeted towards learning linearly separable feature-pairs.

Introduction

What was the problem and its context? (including data sources)
Almeida et al. (2011) produced a work that determined that a support vector machine (SVM) classification model was the recommended solution to classify spam from non-spam messages in SMS text data. The study aggregated data from a wide variety of sources and compared the performance among different configurations of SVMs, naïve bayes, K-NN models and word token patterns.
Why was it a problem?
Despite the resulting models' ability to classify spam from non-spam with an 83.10% accuracy, the token configurations used resulted in an unnecessarily high number of features and failed to consider more complex models that perform well in higher dimensions.
Why was the project necessary and how was the problem solved?
Spam filtering is a process that is increasingly necessary with the widespread use of technology. Unwanted messages not only frustrate users but also strains IT infrastructure and costs businesses billions in lost productivity (Christina et al., 2010). Hence, this work extends previous literature by comparing modern classification models to prior solutions.

Report Body

Methods/AI techniques used

Classification Models

What are classification models?

Classification models are supervised learning techniques that predict one or multiple target categorical variables based on a multi-dimensional input vector (Kari & Amalanathan, 2019).

How are classification models evaluated?

Unlike regression models, classification models do not attempt to predict continuous target variables but rather discrete attributes. This work discusses a binary classification task; thus, it evaluates the predicted outputs (True or False) by evaluating the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) metrics. Each of these metrics contribute to an accuracy or score function outlined in the model evaluations section.

How do they relate to the problem context?

This work aims to predict whether a given SMS text sourced from the work of Almeida et al. (2011) is classified as spam or not.

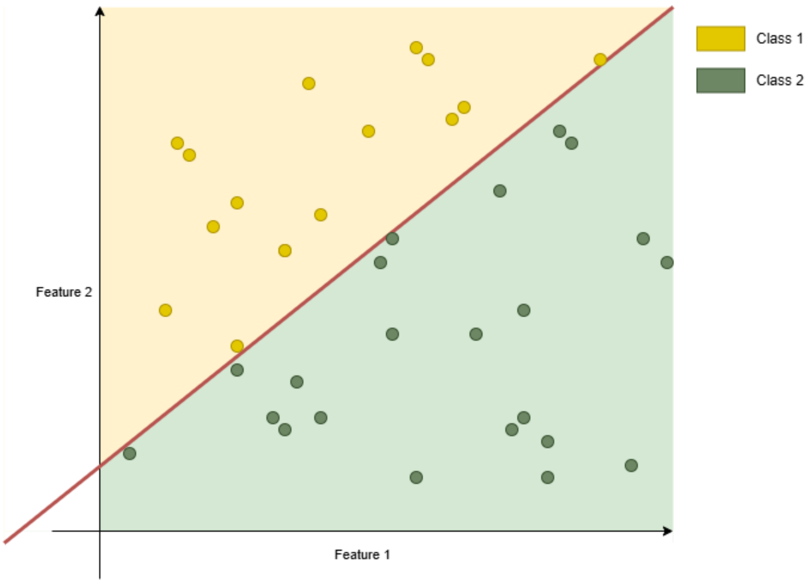

Figure 1: Classification

Neural Network

Description of the technique

Forward Pass

Each input neuron (that is each feature in any instance) is multiplied by a weight and aggregated along with a bias (dot product). Each neuron contains an activation function that places the dot product on a non-linear curve. The result is then parsed into each neuron in the first hidden layer (h0 – hn). The following dot product result is then fed into the next hidden layer and so on. This process continues until a final output (y0 - yn) is produced.

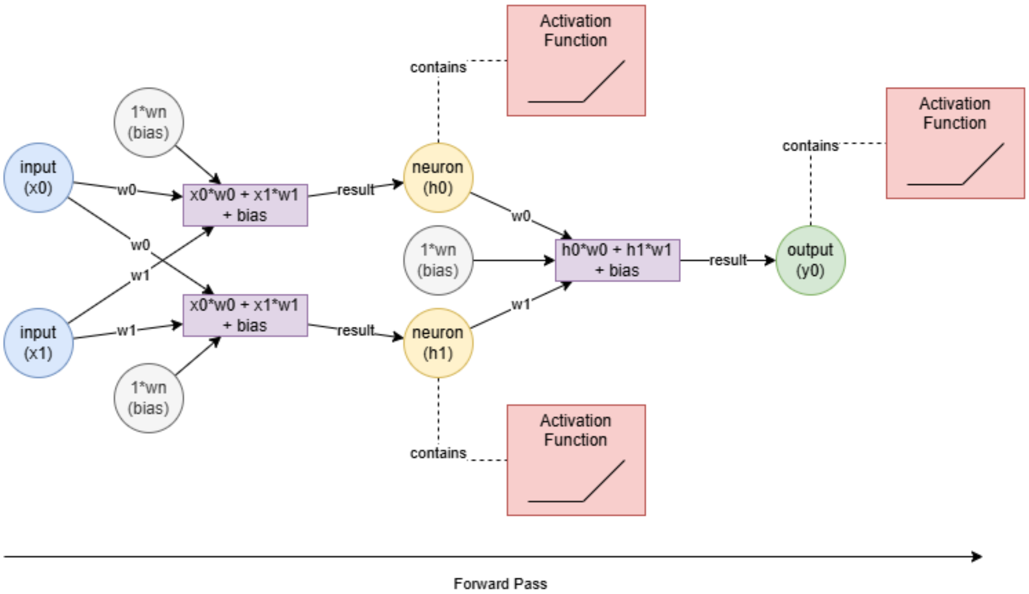

Figure 2: Forward Pass

Why was this algorithm chosen?

Back Propagation

Although an error of 0 is ideal, data mining models are rarely perfect. Hence, to optimize the weights in a neural network, the back propagation process attempts to have the derivative/gradient of the loss function approach 0. In this way, the minimum error can be calculated. This is done by iteratively increasing or decreasing the weights(w) in each neuron composed of the dot product $(w_0h_0 + w_1h_1 + bias)$ and an activation function by the learning rate.

Loss Function: This work uses binary cross entropy loss function for the configured neural network to calculate how accurate the predicted value is to the true value. Intuitively, the cross-entropy function simplifies to the average of each log probability that the true value is labelled as 1.

$Let\ input\ instance = i$

$Let\ desired\ TRUE\ label = y_i$

$Let\ the\ total\ number\ of\ instances/points = N$

$Let\ probability\ of\ desired\ output = p(y_i)$

$\boldsymbol{Cross\ Entropy\ Loss} \rightarrow Loss = \frac{1}{N}\sum_{i=1}^{N} y_i \times \log(p(y_i)) + (1 - y_i) \times \log(1 - p(y_i))$

$Assuming\ the\ TRUE\ label\ is\ 1.$

$Loss = \frac{1}{N}\sum_{i=1}^{N} 1 \times \log(p(1)) + (1 - 1) \times \log(1 - p(1))$

$\boldsymbol{Binary\ Cross\ Entropy\ Loss} \rightarrow Loss = \frac{1}{N}\sum_{i=1}^{N} \log(p(1))$

Gradient Descent: Back propagation iteratively increases or decreases by the learning rate of the weights in each neuron such that the derivative approaches 0 and by extension the minima of the loss function.

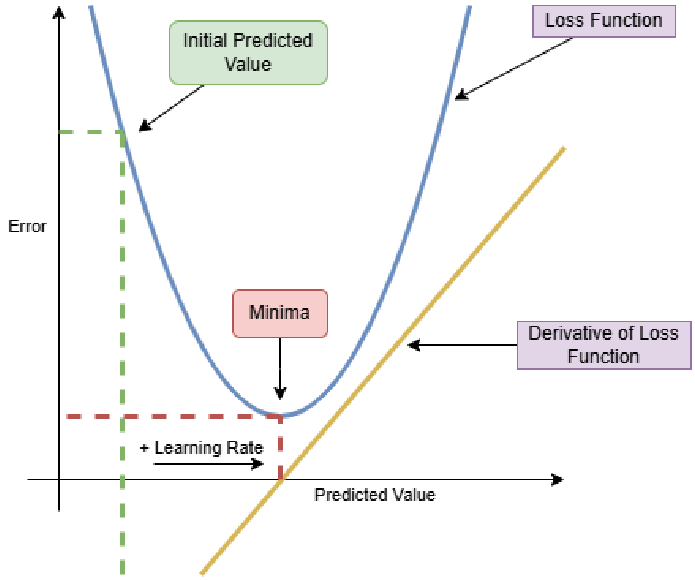

Figure 3: The illustrated loss function purely for explanation. It is not necessarily the shape of the cross-entropy loss function.

Why was this algorithm chosen?

Due to the nature of text classification tasks, the data preprocessing stage often produces high dimensional matrices. Neural networks are applicable to a broad variety of data mining tasks yet were not evaluated in the work of Almeida et al. (2011). This report aims to compare the suitability of feed-forward neural networks for SMS text classification against the recommended models in previous literature.

Support Vector Machine

Description of the technique

Support vector classifiers learn an optimal hyperplane by maximizing the distances between margins. These margins are defined by support vectors that act as the boundaries from one class to another. Typically, classification is predicted using a soft margin where data points may fall between the maximum margins. However, depending on the linear separability of the dataset, hard margins can be used where all data points that fall on one side of the optimal hyperplane are predicted as one class and the remaining points on the other side of the hyperplane are predicted as the other class.

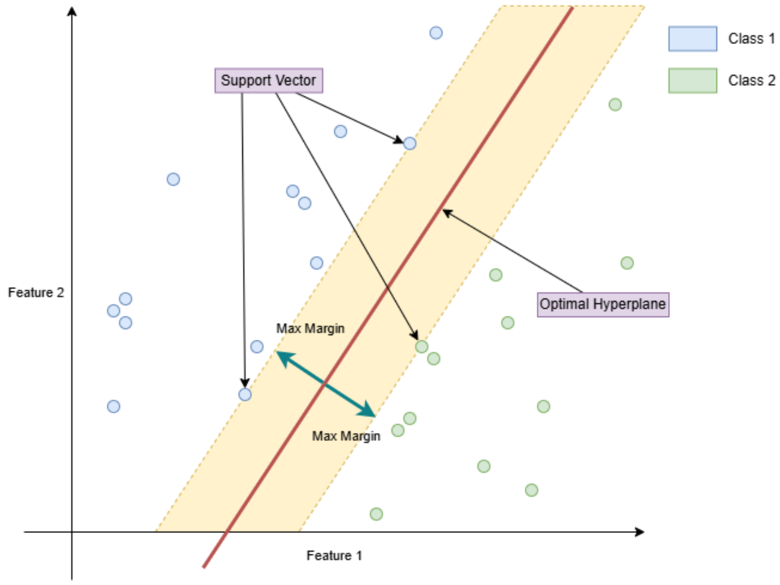

Figure 4: Support Vector Classifier

Why was the algorithm chosen?

The vectorization method in the data preprocessing stage produces a sparse matrix where each unique word in the corpus is represented as its own feature. Thus, the features in the resulting matrix will contain mostly zeros as it is highly unlikely that any given word will appear in all the texts. This makes it such that each feature pair is often linearly separable as the maximal margin spans from zero to some non-zero number. Previous literature recommended the support vector classifier as opposed to alternative models like naïve bayes and KNN classifiers as it accurately predicted 83.10% of spam messages (Almeida et al., 2011).

Implementation of AI techniques

Programming Environment

Variable	Description
CPU	Intel(R) Core(TM) i7-4600M CPU @ 2.90GHz 2.89 GHz
RAM	8.00 GB
Python	Version 3.12.4
	Programming language used for model implementation.
Anaconda	Version 24.5.0
	Jupyter notebooks interface for local testing environment.
Google Colab	Uniform external testing environment for Jupyter notebooks.

Libraries

Package	Usage
Pandas	Data handling library used for slicing and transforming SMS text message data.
Scikit-learn	Used for text vectorization, SVM implementation, and model evaluation.
Tensorflow	Neural network classifier implementation.

Data Source

The dataset used in this report was acquired from the work of Almeida et al. (2011). It contains 5574 rows and two columns: the first containing the message label and the second containing message content. It is identical to the data used in previous literature as a control for model evaluations.

Data Dictionary

Feature	Data Type	Description
Spam	Nominal	Class labels indicating whether the given text message is spam.
Text	Nominal	The text content contained within a text message.

Data Exploration

	Text			
	count	unique	top	freq
Spam				
False	4827	4518	Sorry, I'll call later	30
True	747	653	Please call our customer service representative...	4

Figure: There is a greater distribution of non-spam SMS messages compared to spam messages. Additionally, non-spam messages appear to have a greater frequency of identical messages compared to non-spam messages. This could potentially be attributed to users communicating using short-hand or common phrases.

Data Preprocessing

Tokenization

Tokenization is the method of splitting a sentence into individual words/tokens and removing irrelevant artefacts.

Previous Literature

Almeida et al. (2011) proposed the following tokenization methods:

1. tokens start with a printable character, followed by any number of alphanumeric characters, excluding dots, commas and colons from the middle of the pattern.
2. any sequence of characters separated by blanks, tabs, returns, dots, commas, colons and dashes are considered as tokens.

Criticism

The tokens described in the paper do not adequately capture the words for this data as they are case sensitive, and some unnecessary punctuation is included. The resulting tokens contain redundancies as identical words are coded as separate features in the vectorization process.

<u>Example:</u>

"I was under-prepared."
"I was under-prepared?"
"I was under-prepared!"

Method	Tokens
Tokenizer 1	"was", "under-prepared", "Under-prepared?", "under-prepared!"
Tokenizer 2	"I", "was", "under", "Under", "prepared", "prepared?", "prepared!"

Solution

The scikit-learn library uses a standardized token pattern that matches any sequence of characters of length 2 or greater and excludes all punctuation. The following pattern greatly reduces the number of tokens resulting in fewer dimensions for the vectorization step.

Regular Expression from Scikit-Learn Library:
`` `(?u)\b\w\w+\b` ``
Tokens
“was”, “under”, “prepared”

Vectorization

The process of converting text tokens into numerical feature representations for model training. The resulting dimensions of the vectorized dataset contained 5573 rows and 8713 columns.

Term Frequency Inverse Document Frequency (TF-IDF)

TF-IDF is a probabilistic model of information retrieval where terms that occur in many documents are given lower weights as they are not good discriminators compared words that appear in fewer documents (Robertson, 2004). The weight is calculated by multiplying the term frequency (TF) by the inverse document frequency (IDF).

Component	Description
Term Frequency	The number of occurrences that a token appears. $TF(term, document) = \frac{\#\ occurence\ of\ term\ in\ document}{\#\ total\ words\ in\ document}$
Inverse Document Frequency	A weight for any given token that decreases as the number of occurrences increase. $IDF(term, corpus) = \log\left(\frac{\#\ documents\ in\ corpus}{\#\ documents\ containing\ term}\right)$

Experiments

Support Vector Classifier

Parameter	Configuration	Justification
Kernel	Radial Basis Function	RBF is a popular kernel that is widely used for non-linear classification tasks.

Neural Network

Parameter	Configuration	Justification
Hidden Layers	3	Layers were kept low as the matrix is sparse and likely does not contain many complex relationships between vector pairs.
Neurons	Layer / Neurons x_i 8713 h_0 128 h_1 128 h_2 128 y_i 1	Input and output neurons were determined by the number of features and number of target features respectively. Hidden layer width was initialized from a minimum of 128. It was found that a narrower network made comparable predictions to wider ones. This was likely caused by the sparseness of the vector matrix.
Activation Function	Rectified Linear Unit	Both sigmoid and RELU activation function configurations were tested. However, RELU produced vastly more accurate results (99% accuracy) compared to sigmoid (86% accuracy).
Epochs	10	Reduced number of passes over a single batch as the vector space is sparce resulting is fewer complex relationships. Higher epochs would be inefficient.
Batch Size	512	The dataset contains many rows. Lower batch sizes would cause unnecessarily long computing times.
Loss Function	Binary Cross Entropy	Effective metric for binary classification tasks compared to others like mean square error.
Optimizer	Adam	Efficient method for adjusting the learning rate of the neural network that is a combination of gradient descent and RMSProp optimizer (prakharr0y, 2024).

Train Test Split

This report used only a single train test split. Due to the high dimensionality of the vectorized dataset, the robustness of the study was limited by time and computing constraints.

Parameter	Configuration	Justification
Test Size	0.2	Frequently used split that has a lower risk of overfitting and underfitting compared to more extreme split sizes.

Model Evaluations

Metrics

This paper evaluates accuracy, precision, recall, and f1-score from the scikit-learn classification report. It should be noted that due to the data imbalance of spam and non-spam rows, precision, recall, and f1-score should be priority metrics over accuracy.

Metric	Description
Accuracy	Although accuracy is helpful in model evaluation, it should not be the primary indicator of performance. Typically, it does not perform well when the dataset is not balanced. The SMS text dataset contains 747 instances of spam and 4827 instances of non-spam. $Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$
Precision	Evaluates the accuracy of positive predictions. $Precision = \frac{TP}{TP + FP}$
Recall	Evaluates the sensitivity of true positive rate. $Recall = \frac{TP}{TP + FN}$
F1-score	Combines precision and recall into a single aggregate value. $F1\ Score = 2 \times \frac{Precision * Recall}{Precision + Recall}$

Results

SVM

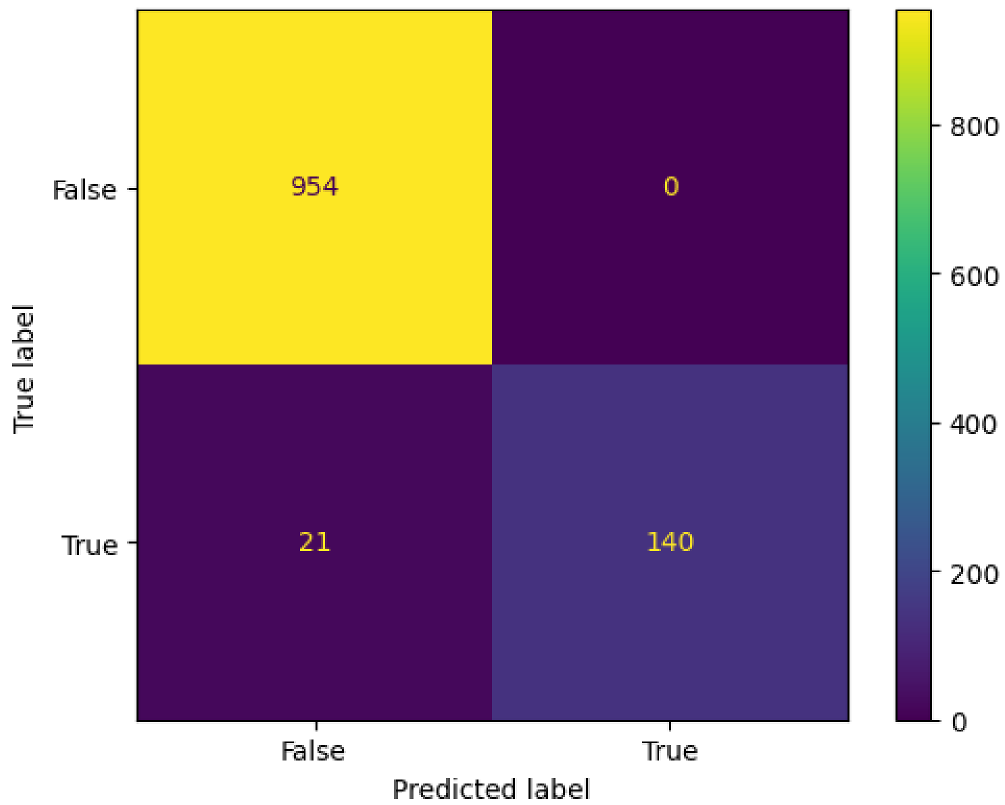

Figure 5: SVM Confusion Matrix

Neural Network

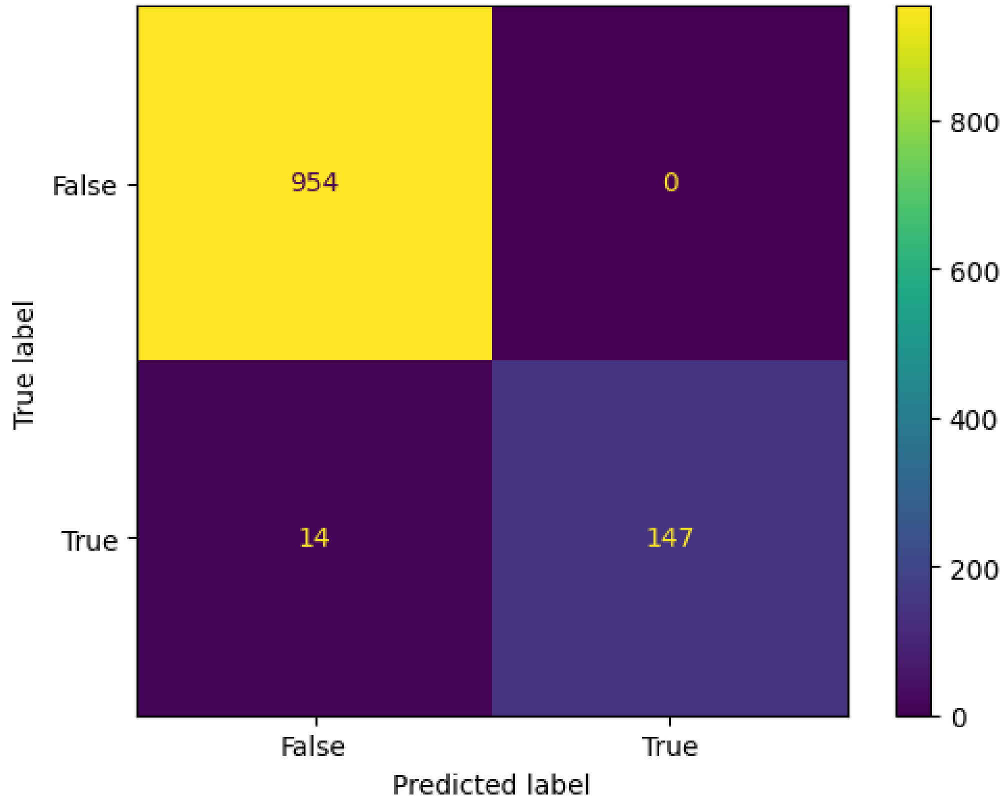

Figure 6: Neural Network Confusion Matrix

Model	F1-Score	Precision	Recall	Accuracy
SVM	0.93	1.00	0.87	0.98
Neural Network	0.95	1.00	0.91	0.99

Findings

For all metrics but precision, the neural network performed better than the SVM classifier. This indicates that for NLP binary classification tasks, neural networks are better at learning the trends in high dimensional vector spaces.

Comparisons of different AI techniques

Support Vector Classifier

Advantages
- Robust to small datasets
- Performance

Disadvantages
- Low scalability
- Difficulty handling high dimensional data

The support vector classifier was able to correctly distinguish between spam and non-spam SMS texts with reasonable accuracy. However, the recall score shows that the SVM classifier predicts a higher number of false negatives comparatively to neural networks.

Due to the nature of NLP tasks and text vectorization, the input sequence usually takes form as a series of features with many dimensions that SVMs generally struggle with.

Neural Network

Advantages
- High complexity handling
- High scalability

Disadvantages
- Data dependency
- Difficult to tune optimal parameters

This work demonstrated that neural networks were able to outperform SVM classifiers for binary NLP classification tasks in precision, recall, f1-score, and accuracy. In the earlier stages of model configuration, it was found that configurations normally used to learn complex non-linear relationships between feature pairs was unnecessary due to the sparsity/separability of the vectors.

Conclusions and Recommendations

Conclusion

The results of this study show that between SVM and neural network classification models, neural networks were able to more accurately classify whether a given SMS text message was spam or non-spam. Due to the sparsity of the vector matrix, the transformed dataset is reasonably separable, reducing the need for neural network configurations targeted towards learning complex non-linear relationships between feature pairs. Hence, further parameter tuning may be able to produce a more computationally efficient model without sacrificing performance. This work expands on previous literature and critiques the recommendation of support vector machines for the use of SMS text classification tasks. Further work into this topic could expand into datasets on social media posts, job postings, etc. Ultimately the task of binary NLP text classification is a broad topic that is applicable to a myriad of disciplines.

Recommendation

When comparing the classification models of neural networks and SVMs, it is recommended that users choose a solution that utilizes some non-complex configuration of a neural network. As shown in the model evaluations section, neural networks outperformed the SVM classifier in all metrics but precision where both models achieved a precision score of 1.00.

List of References

When you use material from other reports or books, you must properly acknowledge it, by citing the report or book in a list of references, and referring to the list at the places in the text where the material is used. Please use The APA (American Psychological Association) referencing style *in this report. Citations and References are important parts of writing in an academic environment.*

Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2011). Contributions to the study of SMS spam filtering: new collection and results. Proceedings of the 11th ACM symposium on Document engineering,

Christina, V., Karpagavalli, S., & Suganya, G. (2010). A study on email spam filtering techniques. *International Journal of Computer Applications*, *12*(1), 0975-8887.

Kari, V., & Amalanathan, G. M. (2019). Synthesis of Classification Models and Review in the Field of Machine Learning. *Advanced Classification Techniques for Healthcare Analysis*, 18-51.

prakharr0y. (2024, 20 Mar, 2024). *What is Adam Optimizer?* Geeks for Geeks. Retrieved 28 Sep, 2024 from https://www.geeksforgeeks.org/adam-optimizer/

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation*, *60*(5), 503-520.

Appendices

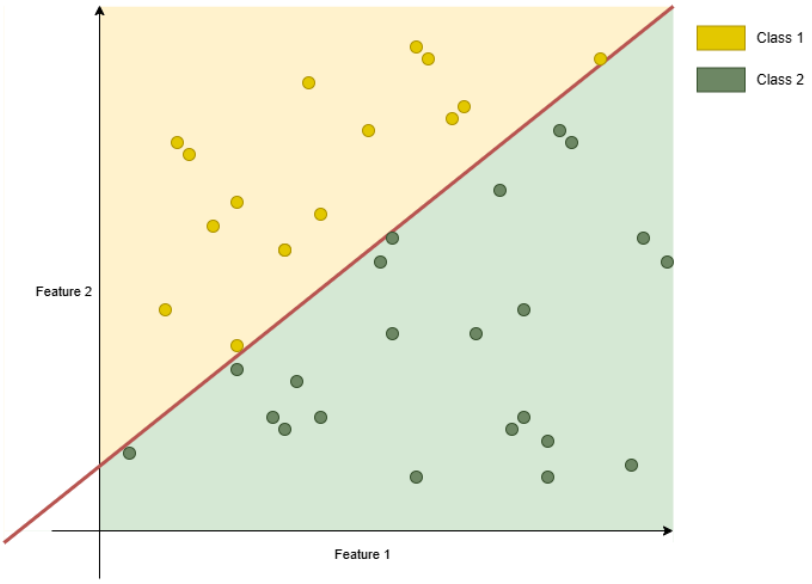

Figure 1: Classification

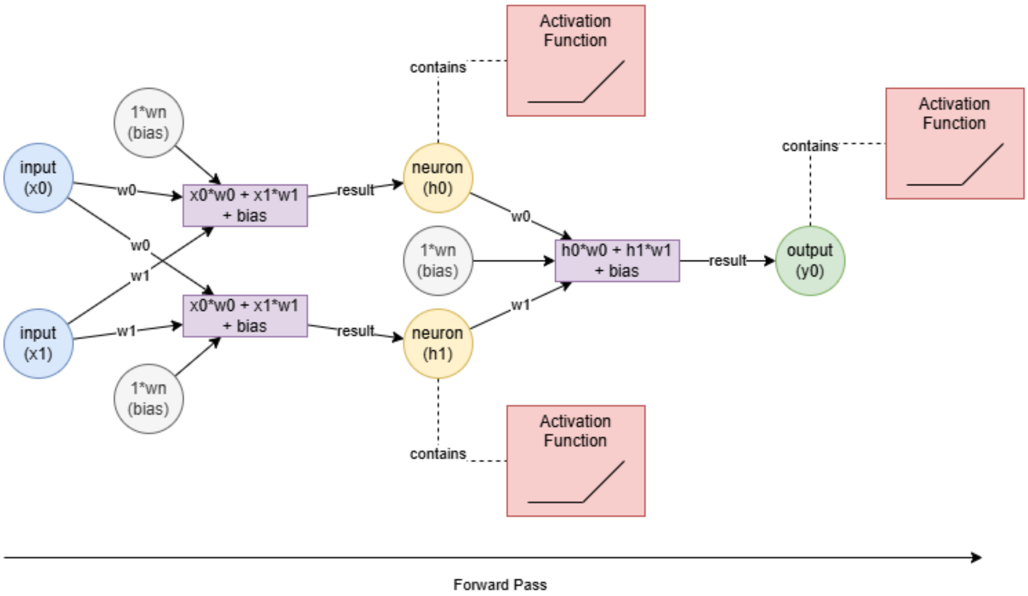

Figure 2: Forward Pass

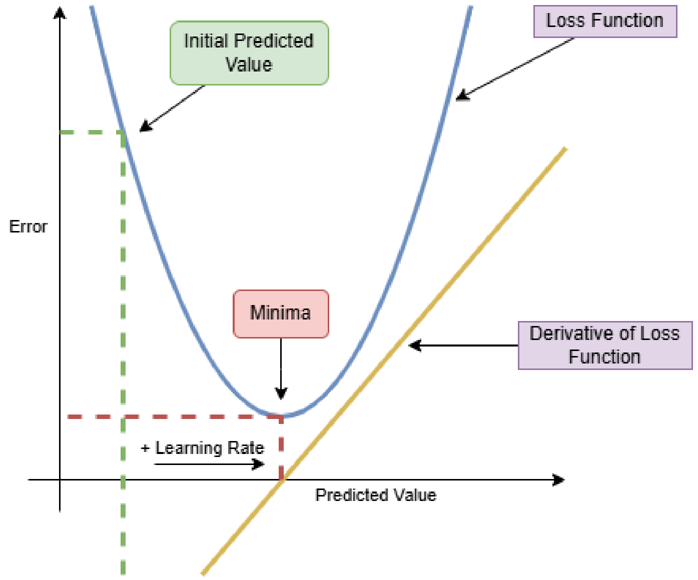

Figure 3: Gradient Descent

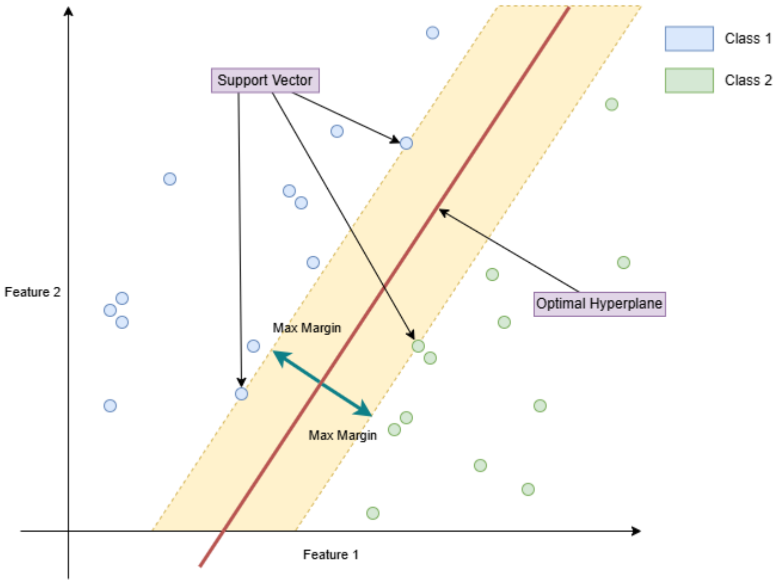

Figure 4: Support Vector Classifier

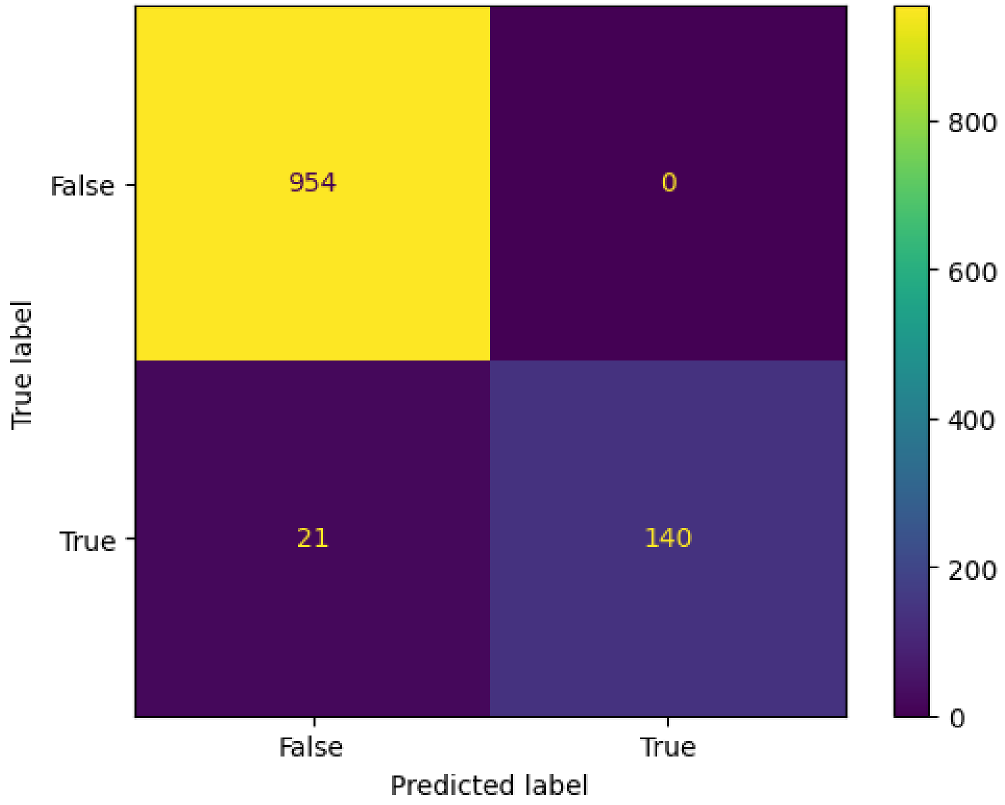

Figure 5: SVM Confusion Matrix

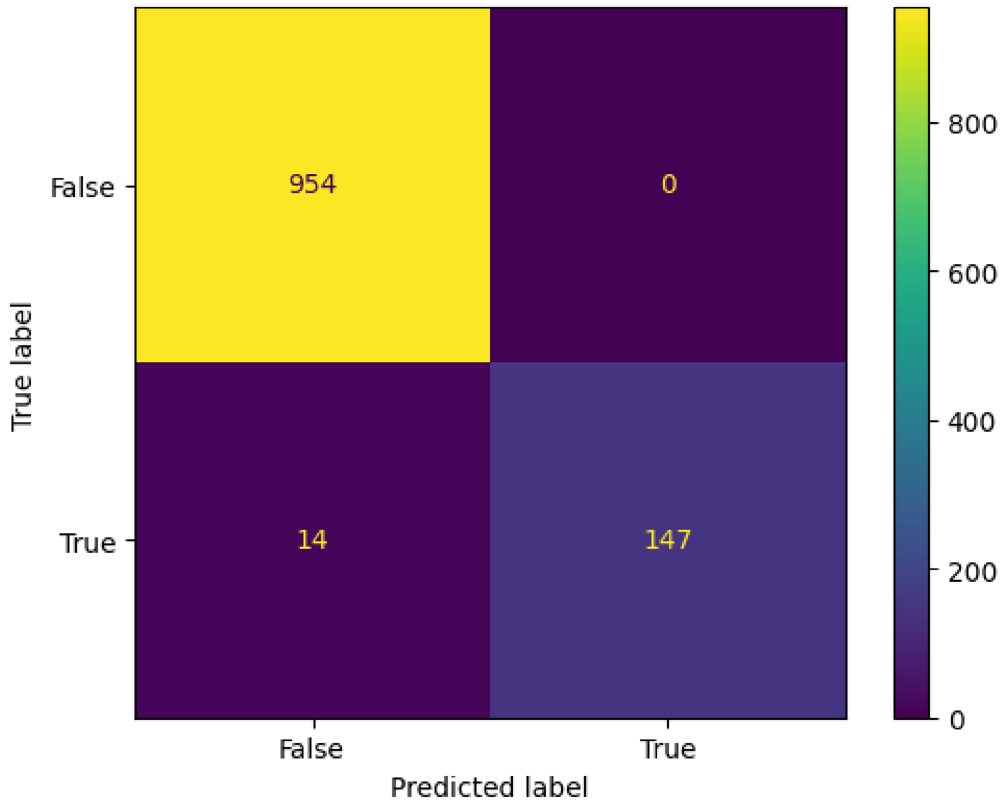

Figure 6: Neural Network Confusion Matrix

Individual reflection

Planning
Identified knowledge gaps and pitfalls in existing literature.
This report did not plan to directly mirror the work of previous literature. Rather it attempts to compare modern solutions to those previously recommended.
Any intuition that did not appear in previous work was reengineered and justified based on personal knowledge and additionally research that was not directly related to the main article.

Learnings
Learned the process/pipeline of text preprocessing into tokens and vectors for classification tasks.
Understood the structure of the resulting sparse matrix from the vectorization process and its effects on model performance.

Challenges
Due to the time constraint of the project, it was difficult to justify using a more robust training and testing method as earlier iterations of the model during the parameter tuning stage required upwards of 20+ minutes to train.
The optimal solution of hyperparameters is still unknown as they were tuned manually.
Was difficult to shadow/mirror previous literature as it did not provide the specific hyperparameters used to reproduce their results.

Limitations
Time constraint prevented robust model evaluation and hyperparameter optimization.
Could not reproduce the results in existing work.

Future
Apply the classification model to different datasets and evaluate the effects/performance.
Vary the size of the dataset to evaluate accuracy for more complex data.
Implement selected models into real-time systems and determine the accuracy of predictions while the system is in production.