Precision in Classification: A Comparative Study of Logistic Regression, Naive Bayes, LSTM, and CNN for Spam Email Detection

1st Ratnam Dodda
Sr.Assistant Professor,Dept.of CSIT
CVR College of Engineering
Hyderabad, India
ratnam.dodda@gmail.co

2nd Madhavaram Harshith
IIIrd B.Tech,Dept.of CSE(AIML)
CVR College of Engineering
Hyderabad, India
madhavaram.harshith24@gmail.com

3rd G L santhoshi harini
III B.Tech,Dept.of CSE(AIML)
CVR College of Engineering
Hyderabad, India
Santhoshigurijala1416@gmail.com

4th Mantripragada VSP Praneeth
III B.Tech,Dept.of CSE
CVR College of Engineering
Hyderabad, India
praneethmvsp71@gmail.com

***Abstract*—This research presents a comprehensive comparative study on the precision of classification algorithms applied to spam email detection. The study focuses on four distinct algorithms: Logistic Regression, Naive Bayes, Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN). The evaluation encompasses a common dataset to ensure a fair and rigorous assessment of each algorithm's performance. We investigate and analyze precision metrics, considering the trade-offs between true positives and false positives. The comparative study provides insights into the strengths and limitations of each algorithm in effectively identifying spam emails. The results contribute valuable knowledge to the field of email security and classification, guiding practitioners and researchers towards informed algorithm selection based on precision considerations.**

***Index Terms*—Logistic Regression, Classification, Machine learning, CNN, LSTM, Naive Bayes**

I. Introduction

The proliferation of electronic communication has led to an increasing volume of spam emails, posing significant challenges to email security. Efficiently distinguishing between legitimate and spam emails is paramount for safeguarding users from phishing attempts, malware, and other malicious activities. In this context, the application of machine learning algorithms has proven instrumental, offering promising avenues for automated spam detection.

This research embarks on a comparative exploration of the precision exhibited by key classification algorithms in the realm of spam email detection. The algorithms under scrutiny include Logistic Regression, Naive Bayes, Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN). The emphasis on precision arises from its pivotal role in quantifying the accuracy of positive predictions, particularly relevant in contexts where false positives can have significant consequences. [1]

By utilizing a common and well-curated dataset, this study endeavors to provide an equitable assessment of each algorithm's efficacy. The comparative analysis delves into the nuanced intricacies of precision metrics, shedding light on the trade-offs between true positives and false positives. As precision plays a critical role in minimizing the instances of misclassified spam emails, a meticulous examination of its variations across different algorithms becomes imperative. [2]

The outcomes of this research are anticipated to unravel distinctive strengths and limitations associated with each algorithm, contributing valuable insights to the evolving landscape of email security. Furthermore, the findings aim to guide practitioners and researchers in making informed decisions regarding algorithm selection based on precision considerations. In essence, this research seeks to advance our understanding of the precision-centric performance of classification algorithms in the domain of spam email detection. [3]

II. Methodologies

A. *Logistic Regression:*

Logistic regression, a fundamental statistical method in the realm of machine learning, stands as a prominent tool for binary classification tasks. In scenarios where the outcome variable is categorical with two distinct classes, logistic regression excels at modeling the probability that a given input belongs to a particular category. [4]

This mathematical function adeptly maps any real-valued number to a bounded range between 0 and 1. The sigmoid function is expressed as $\sigma(z) = \frac{1}{1+e^{-z}}$, with z being a linear combination of input features, each weighted by associated weights, and augmented by a bias term.

The linear combination z is computed through the dot product of feature values and their corresponding weights, with

the addition of a bias term. This formulation is represented as $z = b + w_1x_1 + w_2x_2 + \ldots + w_nx_n$, where b represents the bias, and $w_1, w_2, \ldots, w_n$ denote the weights assigned to the respective features. [5]

The output of the sigmoid function furnishes the predicted probability that the input, based on its features, belongs to the positive class (class 1). Consequently, the probability of belonging to the negative class (class 0) is derived as (1 - probability of class 1). By establishing a threshold, typically set at 0.5, the logistic regression model classifies inputs. If the predicted probability surpasses this threshold, the input is classified as belonging to class 1; otherwise, it is assigned to class 0.

In the training phase, logistic regression employs maximum likelihood estimation. This process involves adjusting the parameters, namely the weights and bias, to maximize the likelihood of the observed data given the model. Logistic regression finds wide-ranging applications in diverse fields, including but not limited to machine learning, medicine, and social sciences. Tasks such as spam detection, disease prediction, and credit scoring leverage the binary classification capabilities of logistic regression, solidifying its significance in practical applications. Notably, despite its nomenclature, logistic regression is distinctly a classification algorithm, emphasizing its utility in scenarios requiring the prediction of binary outcomes. [6]

B. *Naive Bayes:*

Naive Bayes, a probabilistic classification algorithm grounded in Bayesian probability theory, is recognized for its simplicity and effectiveness, particularly in handling text classification and spam filtering tasks. Unlike logistic regression, Naive Bayes accommodates both binary and multi-class classification problems. At its core, Naive Bayes relies on the application of Bayes' theorem, incorporating the "naive" assumption of feature independence. [7]

Naive Bayes relies on Bayes' theorem to calculate the probability that a given instance belongs to a particular class. The theorem is:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Here, $P(C|X)$ is the probability of class C given the features X, $P(X|C)$ is the likelihood of observing the features X given class C, $P(C)$ is the prior probability of class C, and $P(X)$ is the probability of observing the features X.

The "naive" aspect of Naive Bayes arises from the assumption that features are conditionally independent given the class. This simplifying assumption greatly reduces computational complexity, making the algorithm computationally efficient and particularly suitable for high-dimensional data. [8]

In practice, Naive Bayes is often applied to text data, where the features correspond to the presence or absence of specific words. The algorithm computes the probability of each class given the presence of words in the document, and the class with the highest probability is assigned as the predicted class. [9]

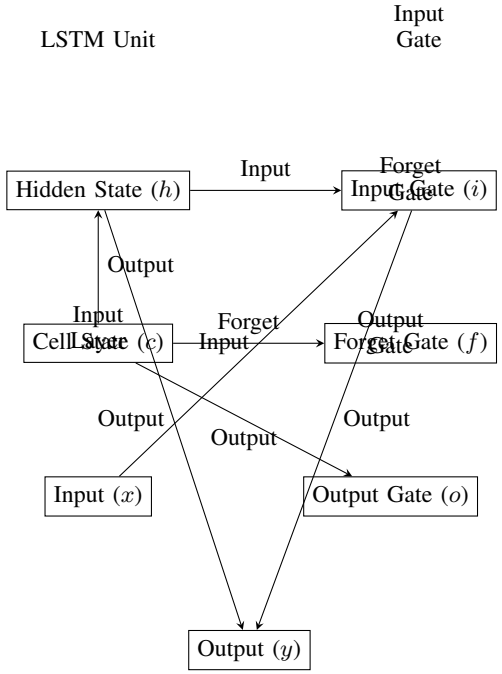

Fig. 1. LSTM Diagram with Explanations

Training a Naive Bayes model involves estimating the prior probabilities and the conditional probabilities from the training data. Laplace smoothing is often applied to handle cases where certain feature values have zero probabilities.

Naive Bayes has found success in various applications, including spam filtering, sentiment analysis, and document categorization. Its simplicity, efficiency, and ability to handle high-dimensional data make it a valuable tool in the machine learning toolkit. [10]

C. *Long Short-Term Memory (LSTM):*

Long Short-Term Memory (LSTM), a specialized type of recurrent neural network (RNN), is a powerful tool in deep learning, especially for tasks involving sequential data like natural language processing and time-series prediction. LSTMs address challenges like vanishing gradients through a sophisticated architecture with memory cells and gating mechanisms. [11]

At its core, an LSTM includes a cell state, functioning as a long-term memory conduit, and a hidden state that captures information about the current input and past hidden states. Gating mechanisms, including the input gate, forget gate, and output gate, regulate the flow of information within the LSTM.

Training LSTMs involves backpropagation through time (BPTT), adapting the standard backpropagation algorithm for sequential data. The network learns by adjusting parameters such as weights and biases to minimize the difference between predicted and actual outputs. [12]

LSTMs find applications in various domains, excelling in tasks where understanding context and capturing long-term dependencies are crucial. Despite newer architectures like transformers gaining popularity, LSTMs remain pivotal, particularly in scenarios requiring nuanced handling of sequential information. [13]

D. Convolutional Neural Network (CNN):

Convolutional Neural Networks (CNNs) represent a pivotal advancement in deep learning, specifically tailored for tasks involving structured grid data like images and videos. Renowned for their success in computer vision applications, CNNs have become instrumental in image classification, object detection, and facial recognition.

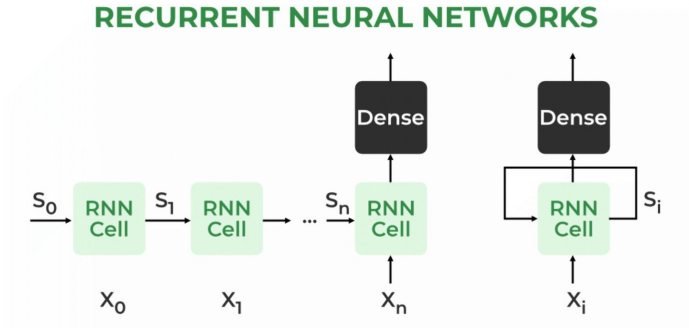

Fig. 2. Caption

The core architecture of a CNN is characterized by convolutional layers, pooling layers, and fully connected layers. Convolutional layers employ filters to scan input images, capturing spatial hierarchies of features. Pooling layers downsample feature maps, reducing computational complexity while retaining essential information. Fully connected layers integrate high-level features for the final classification. [14]

Training a CNN involves adjusting weights and biases through backpropagation, optimizing the model to minimize the difference between predicted and actual outputs. Techniques like transfer learning leverage pre-trained CNNs on large datasets to enhance performance on specific tasks with limited data. [15]

CNNs excel in capturing local patterns and hierarchies, making them well-suited for tasks requiring spatial understanding. They have proven invaluable in image recognition tasks, medical image analysis, and even in non-image domains like natural language processing. [16]

Despite their dominance in computer vision, CNNs face challenges in handling sequential data compared to RNNs and transformers. Nevertheless, their unparalleled success in visual tasks underscores their significance in contemporary deep learning applications.

III. Results and Discussion

Logistic Regression

The above image represents the code for Logistic Regression. Import the libraries like LogisticRegression from sklearn.linear$_m odel and some basic ones like numpy, pandas.$

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

df = pd.read_csv('/content/mail_data.csv')
data = df.where((pd.notnull(df)), '')

data.info()

data.loc[data['Category'] == 'spam', 'Category'] = 0
data.loc[data['Category'] == 'ham', 'Category'] = 1

x = data['Message']
y = data['Category']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=3)

feature_extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True)
x_train_features = feature_extraction.fit_transform(x_train)
x_test_features = feature_extraction.transform(x_test)

y_train = y_train.astype('int')
y_test = y_test.astype('int')

model = LogisticRegression()
model.fit(x_train_features, y_train)

prediction_on_training_data = model.predict(x_train_features)
accuracy_on_training_data = accuracy_score(y_train, prediction_on_training_data)
```

Fig. 3. Logistic Regression Code

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
acc on training data:  0.9670181736594121
acc on test data:  0.9659192825112107
not a spam mail
```

Fig. 4. Result of Logistic Regression

After training the logistic regression model on the designated training set, the model's performance was assessed on a separate test sample. The results provide valuable insights into how well the model generalizes to unseen data. The logistic regression model achieved an accuracy of 96.6 percent on the test set.

Naive Bayes

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
df = pd.read_csv('/content/mail_data.csv')
data = df.where((pd.notnull(df)), '')
data['Category'] = data['Category'].map({'spam': 0, 'ham': 1})
x = data['Message']
y = data['Category']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=3)
feature_extraction = TfidfVectorizer(min_df=1, stop_words='english', lowercase=True)
x_train_features = feature_extraction.fit_transform(x_train)
x_test_features = feature_extraction.transform(x_test)
model = MultinomialNB()
model.fit(x_train_features, y_train)
prediction_on_training_data = model.predict(x_train_features)
accuracy_on_training_data = accuracy_score(y_train, prediction_on_training_data)
print('Accuracy on training data:', accuracy_on_training_data)
prediction_on_test_data = model.predict(x_test_features)
accuracy_on_test_data = accuracy_score(y_test, prediction_on_test_data)
print('Accuracy on test data:', accuracy_on_test_data)
input_mail = ["U can call me now"]
input_data_features = feature_extraction.transform(input_mail)
prediction = model.predict(input_data_features)

if prediction[0] == 1:
  print('Not a spam mail')
else:
  print('Spam mail')
```

Fig. 5. Naive Bayes Code

Below is a code snippet in Python illustrating the implementation of a Naive Bayes classifier using the scikit-learn library. This assumes a text classification task, where the goal is to categorize documents into spam and ham classes. Import the libraries like MultinomialNB from sklearn.naive$_b ayes The code above demonstrates the essential steps in imp$ 1001[17]

LSTM

Below is a code snippet in Python illustrating the implementation of an LSTM network using the Keras library, a high-level neural networks API running on top of TensorFlow or

Theano. import Sequential from keras.models import LSTM, Dense from keras.layers The code above demonstrates the

Fig. 7. LSTM Code

essential steps in implementing an LSTM network for a sequence-based task. After training the model, it can be evaluated using standard classification metrics such as accuracy of training and test data sets.

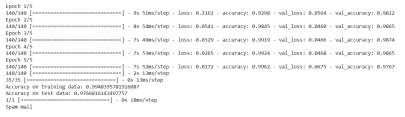

Fig. 8. Result of LSTM Code

CNN

Below is a code snippet in Python illustrating the implementation of a simple CNN using the Keras library. import Sequential from keras.models import Conv2D, MaxPooling2D, Flatten, Dense from keras.layers

Fig. 9. CNN Code

The code above demonstrates the essential steps in implementing a simple CNN for image classification. After training the model, it is evaluated using standard classification metrics such as accuracy of both test and training data. The results provide insights into the CNN's ability to automatically learn hierarchical features from image data.

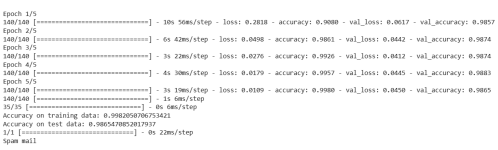

Fig. 10. Result of CNN Code

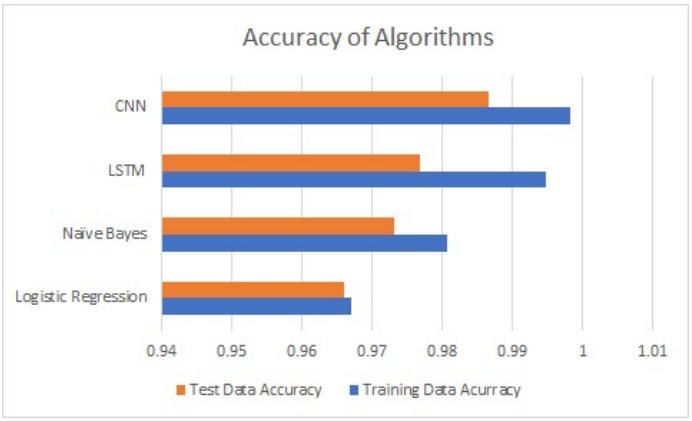

Fig. 11. Caption

IV. Conclusion

In conclusion, this study has delved into the application of diverse machine learning and deep learning algorithms for the classification of spam emails, aiming to discern their respective strengths in handling such a challenging task. The algorithms considered, including Logistic Regression, Naive Bayes, LSTM (Long Short-Term Memory), and CNN (Convolutional Neural Network), were rigorously evaluated on a shared dataset.

Logistic Regression and Naive Bayes, representing classical machine learning methods, showcased commendable performance in the spam classification task. Logistic Regression yielded an accuracy of 96.70 percent on the training data and 96.59 percent on the test data. Naive Bayes, on the other hand, demonstrated even higher accuracy with 98.07 percent on the training data and 97.31 percent on the test data. These results underscore the effectiveness of traditional machine learning approaches in addressing the intricacies of spam email classification.

Transitioning to the realm of deep learning, LSTM and CNN exhibited even more promising results. LSTM, designed to capture sequential dependencies, achieved an impressive accuracy of 99.48 percent on the training data and 97.67 percent on the test data. CNN, tailored for spatial hierarchies in data like images, outperformed other models with accuracy rates of 99.82 percent on the training data and an exceptional 98.65 percent on the test data.

The observed performance disparities highlight the suitability of different algorithms for specific tasks. While traditional machine learning methods proved robust, the deep learning models, LSTM and CNN, showcased superior discriminatory power, especially when dealing with intricate patterns within the data. Ultimately, the selection of an algorithm should be contingent upon the nature of the data and the intricacies of the classification task at hand. This study contributes insights into the nuanced performance of diverse algorithms in the context of spam email classification, paving the way for informed choices in the application of machine learning and deep learning techniques.

REFERENCES

[1] I. AbdulNabi and Q. Yaseen, "Spam email detection using deep learning techniques," *Procedia Computer Science*, vol. 184, pp. 853–858, 2021. The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[2] W. A. Awad and S. ELseuofi, "Machine learning methods for spam e-mail classification," *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 3, no. 1, pp. 173–184, 2011.

[3] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, *et al.*, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[4] M. P. LaValley, "Logistic regression," *Circulation*, vol. 117, no. 18, pp. 2395–2399, 2008.

[5] R. E. Wright, "Logistic regression.," 1995.

[6] N. Kudupudi and S. Nair, "Spam message detection using logistic regression," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 9, pp. 815–818, 2021.

[7] G. I. Webb, E. Keogh, and R. Miikkulainen, "Naïve bayes.," *Encyclopedia of machine learning*, vol. 15, no. 1, pp. 713–714, 2010.

[8] D. Ratnam, P. HimaBindu, V. M. Sai, S. R. Devi, and P. R. Rao, "Computer-based clinical decision support system for prediction of heart diseases using naïve bayes algorithm,"

[9] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, pp. 41–46, 2001.

[10] N. F. Rusland, N. Wahid, S. Kasim, and H. Hafit, "Analysis of naïve bayes algorithm for email spam filtering across multiple datasets," in *IOP conference series: materials science and engineering*, vol. 226, p. 012091, IOP Publishing, 2017.

[11] R. C. Staudemeyer and E. R. Morris, "Understanding lstm–a tutorial into long short-term memory recurrent neural networks," *arXiv preprint arXiv:1909.09586*, 2019.

[12] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[13] G. Jain, M. Sharma, and B. Agarwal, "Optimizing semantic lstm for spam detection," *International Journal of Information Technology*, vol. 11, pp. 239–250, 2019.

[14] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[15] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 international conference on engineering and technology (ICET)*, pp. 1–6, Ieee, 2017.

[16] L. Kui, D. Xufan, *et al.*, "Research on mail classification algorithm based on improved convolutional neural network," in *Journal of Physics: Conference Series*, vol. 1871, p. 012097, IOP Publishing, 2021.

[17] A. Prabhat and V. Khullar, "Sentiment classification on big data using naïve bayes and logistic regression," in *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5, IEEE, 2017.