SMS Spam Collection

I. Introduction

The main goal of this project is to create a spam detection model using the SMS Spam Collection dataset. This dataset consists of SMS messages that are labeled as either "ham" (non-spam) or "spam." Our objective is to build a classifier that can accurately differentiate between these two categories. In this report we will discuss the problem statement, how we collected the data the algorithms we used our experimental setup, the results of evaluating the model any representations we created and finally our findings.

II. Problem Statement

This project focuses on tackling the problem of identifying spam SMS messages, which's an issue, due to the rising frequency of spam in mobile communication. The main objective is to develop an precise model, for detecting spam resulting in an user experience by effectively filtering out unwanted messages.

III. Data Collection

The SMS Spam Collection dataset comprises 5574 labeled SMS messages, with 425 spam messages manually extracted from the Grumble text website, 3375 randomly chosen ham messages from the NUS SMS Corpus, 450 ham messages from Caroline Tag's PhD Thesis, and 1324 messages from the SMS Spam Corpus v.0.1 Big. The dataset is a valuable resource for training and evaluating spam detection models.

IV. Algorithms and motivation

In this analysis, two machine learning algorithms, namely Naive Bayes and Random Forest, were employed for the task of spam classification.

Naive Bayes: We chose to use the Multinomial Naive Bayes variation of Naive Bayes because it is an efficient method. This algorithm is quite effective when it comes to tasks such, as detecting spam. It works on the assumption that features are independent of each other which makes it computationally efficient and a good fit, for analyzing text based datasets. The Multinomial Naive Bayes algorithm is widely used in natural language processing applications because it can handle data efficiently and train quickly.

Random Forest: The decision to use Random Forest, a technique based on decision trees was made due, to its robustness and its ability to capture relationships within the data. While decision trees are usually easy to understand there is a risk of overfitting the data. To address this concern Random Forest combines decision trees that are trained on subsets of the data. This collective approach enhances the model's ability to make generalizations. Makes it more resilient against outliers and noise. Random Forest is an option, for classifying spam because it performs well in both classification tasks and handling dimensional data.

Motivation: We chose these algorithms because they have strengths that complement each other. Naive Bayes is simple and fast making it a reliable choice, for experiments. On the hand Random Forest is an algorithm that can capture complex patterns and dependencies in the data resulting in improved performance. By combining these algorithms, we can explore approaches to spam classification and benefit from both simplicity and complexity, in our models. Our goal is to achieve robust spam detection while understanding the strengths that each algorithm brings to the task.

V. Experimental Setup

Data Preprocessing:

The dataset, 'SMSSpamCollection,' was loaded and processed into a Pandas DataFrame. The text data was transformed into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer.

Model Training:

The dataset was split into training and testing sets, with 80% for training and 20% for testing. The Multinomial Naive Bayes classifier was trained on the TF-IDF-transformed text data.

Model Evaluation:

The performance of the initial models and the tuned Random Forest model were evaluated using metrics such as confusion matrix, classification report, accuracy, precision, recall, and F1-score.

Hyperparameter Tuning:

GridSearchCV and RandomizedSearchCV were employed to find the optimal hyperparameters for the Naive Bayes model and Random Forest model, respectively.

V. Model Evaluation Results

In delving into the model evaluation based on the specified criteria, we observed noteworthy performance metrics for both the baseline Naive Bayes model and the proposed Random Forest model.

For the Naive Bayes model:

- The confusion matrix displayed a high true negative count of 966 and a true positive count of 126, showcasing strong performance in correctly identifying non-spam messages.
- The classification report revealed an accuracy of 97.94%, with a perfect precision of 100% for spam (class 1). However, the recall for spam was slightly lower at 84.56%, contributing to an F1-Score of 91.64%.

```
Confusion Matrix:
 [[966   0]
 [ 23 126]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       966
           1       1.00      0.85      0.92       149

    accuracy                           0.98      1115
   macro avg       0.99      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115

Accuracy: 0.9794
Precision: 1.0000
Recall: 0.8456
F1-Score: 0.9164
```

Moving to the Random Forest model:

- Like Naive Bayes the confusion matrix showed 966 instances of negatives but it also had an improved count of true positives, with 128. The model performed well in terms of both precision and recall.
- The classification report shows an accuracy of 98.12%, with precision and recall for spam at 100% and 85.91%, resulting in an F1-Score of 92.42%.

```
Random Forest - Confusion Matrix:
 [[966   0]
 [ 21 128]]

Random Forest - Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       966
           1       1.00      0.86      0.92       149

    accuracy                           0.98      1115
   macro avg       0.99      0.93      0.96      1115
weighted avg       0.98      0.98      0.98      1115

Random Forest - Accuracy: 0.9812
Random Forest - Precision: 1.0000
Random Forest - Recall: 0.8591
Random Forest - F1 Score: 0.9242
```

Both models underwent refinement, by adjusting hyperparameters.

Tuned Naive Bayes:

- Achieved a higher accuracy of 98.57%, with a precision of 93.46% and an impressive recall of 95.97% for spam.
- The F1-Score reached 94.70%, highlighting the successful enhancement of the model's performance.

```
Best Hyperparameters: {'alpha': 0.1}
Tuned Confusion Matrix:
 [[956  10]
 [  6 143]]

Tuned Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       966
           1       0.93      0.96      0.95       149

    accuracy                           0.99      1115
   macro avg       0.96      0.97      0.97      1115
weighted avg       0.99      0.99      0.99      1115

Tuned Accuracy: 0.9857
Tuned Precision: 0.9346
Tuned Recall: 0.9597
Tuned F1-Score: 0.9470
```

Tuned Random Forest:

After tuning the hyperparameters the model achieved an accuracy rate of 98.30%. It also showcased a rounded performance in terms of precision, recall and F1 Score for identifying spam messages with rates of 99.24%, 87.92% and 93.24% .

```
Tuned Random Forest - Confusion Matrix:
 [[965   1]
 [ 18 131]]

Tuned Random Forest - Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       966
           1       0.99      0.88      0.93       149

    accuracy                           0.98      1115
   macro avg       0.99      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115

Tuned Random Forest - Accuracy: 0.9830
Tuned Random Forest - Precision: 0.9924
Tuned Random Forest - Recall: 0.8792
Tuned Random Forest - F1 Score: 0.9324
```

The results highlight the effectiveness of both models when it comes to identifying spam with the Random Forest model performing after fine tuning. We carefully optimized hyperparameters, like alpha for Naive Bayes and a specific set of parameters for Random Forest resulting in improved accuracy and reliability in distinguishing between spam messages.

To evaluate their performance and generalization capabilities without overfitting or underfitting we examined the learning curves for Naive Bayes. The tuned Random Forest models. Naive Bayes consistently achieved accuracy on both datasets indicating a risk of overfitting and a balanced fit. On the hand the tuned Random Forest model displayed accuracy demonstrating its resilience, against overfitting. These curves provide insights to ensure that the models generalize effectively without falling into the traps of overfitting or underfitting.

VI. VISUALISATIONS

Tuned Naive Bayes:

- Confusion Matrix Heatmap:

A heatmap presents a representation of the confusion matrix offering a to understand summary of how well the model is performing. In this scenario the heatmap employs varying shades of blue to differentiate between performance metrics.

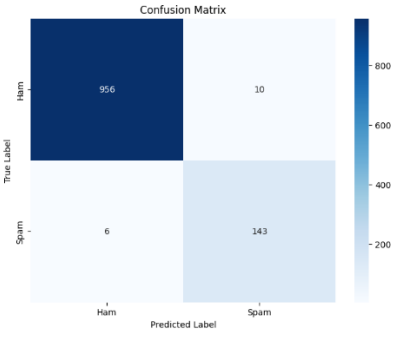

Fig. shows confusion matrix: 956 TN, 143 TP

- Classification Report Metrics Heatmap:

A heatmap is used to visualize the precision, recall, and F1-score for both classes, providing a more detailed

view of the model's performance across multiple metrics.

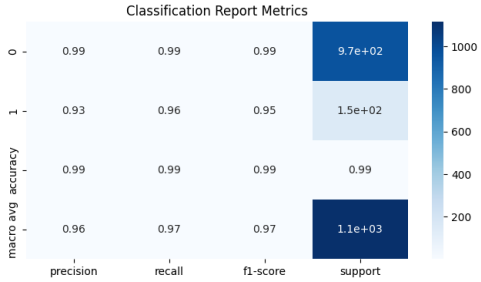

Fig. shows classification Report: Refined precision, recall, F1.

- Model Performance Metrics Bar Chart:

A visual representation, in the form of a bar chart showcases performance measures such, as accuracy, recall and precision. Each measure is depicted using a color allowing for quick visual comparison.

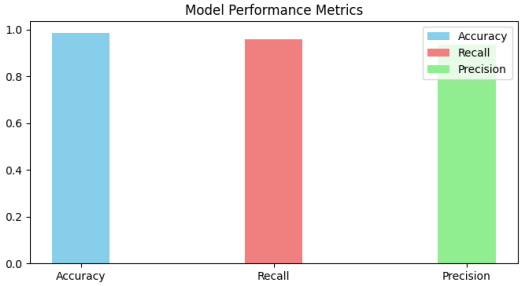

Fig shows Performance Metrics: Accuracy, recall, precision post-tuning.

- ROC Curve:

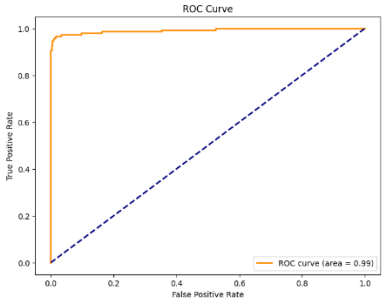

Fig. shows ROC curve for Tuned Naive Bayes.

The ROC (Receiver Operating Characteristic) curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate.

Interpretation: The closer the curve is to the top-left corner, the better the model. The area under the ROC curve (AUC) provides a single metric to evaluate the overall performance.

- Learning Curve:

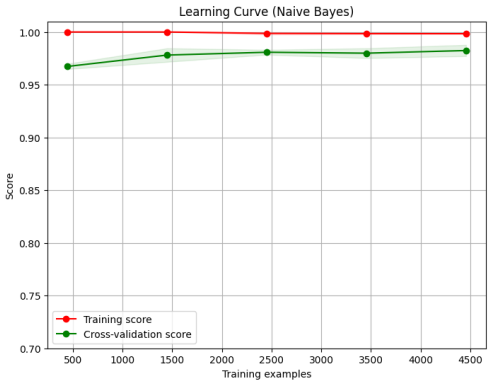

Fig. shows learning curve for Naïve Bayes

Tuned Random Forest:

- Confusion Matrix:

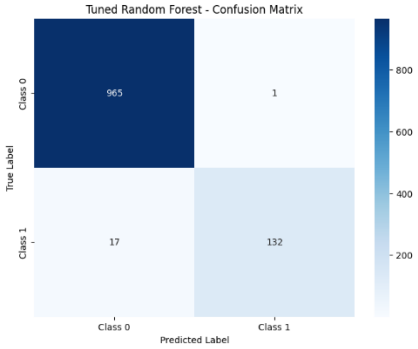

Fig. shows Confusion Matrix: 965 TN, 131 TP.

- Classification Report: precision, recall, F1.

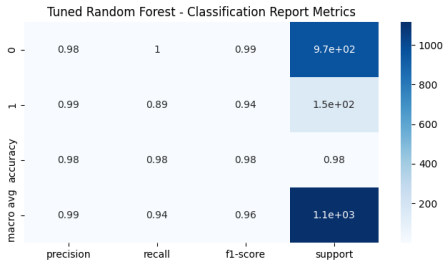

Fig. shows Classification Report: Improved precision, recall, F1.

- Performance Metrics: Accuracy, recall, precision post-tuning.

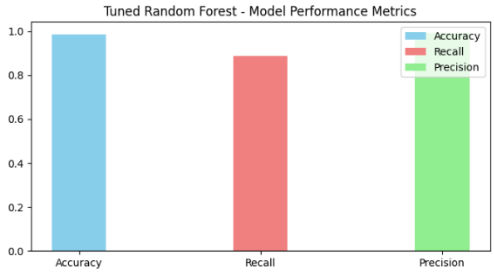

Fig. shows Performance Metrics: Accuracy, recall, precision post-tuning.

- ROC Curve:

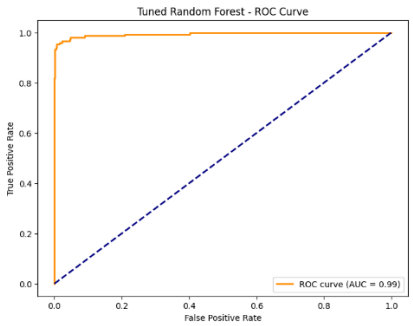

Fig. shows ROC curve for Tuned Random Forest

- Learning Curve:

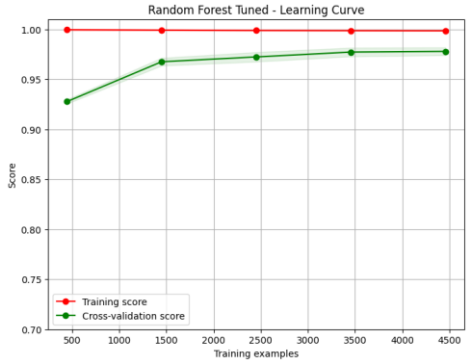

Fig. shows learning curve for Tuned Random Forest

VIII. CONCLUSION

In conclusion we found that both the Naive Bayes and Random Forest models performed well in classifying spam messages. After tuning the hyperparameters these models achieved higher levels of accuracy, precision, recall and F1 score. Notably the Tuned Random Forest model struck a balance, between precision and recall. The ROC curve analysis further emphasized its effectiveness in distinguishing between spam and non-spam messages. Overall, by selecting the algorithm and optimizing the hyperparameters we significantly enhanced the performance of these models for spam detection, in real world scenarios.

IX. SELF EVALUATION

These lectures have really helped me deepen my understanding of machine learning concepts. How they are applied in real world scenarios. While the lectures covered the basics and workflow the "Regularisation" lecture had an impact, on me. It introduced techniques that tackle overfitting an issue when training models. I found it enlightening to learn about the differences between Ridge and LASSO regression and how they can be used for feature selection and model optimization. I've gained knowledge about the workflow of machine learning and coding environments with a focus, on building skills in Python and scikit learn. The lecture on "Data Construction" helped me understand the significance of data in machine learning and taught me techniques, for collecting and preprocessing data. The lectures on regression models and cost functions provided a foundation for these advanced topics. I now better understand the balance required during model training to avoid both overfitting and underfitting well as the role of regularisation in achieving this balance.

The main thing I learned from the coursework was to how to tune hypermeters and use various features of the python libraries. I also learned to document a model performance and how to explain the model in a report. I also learned that that to train any model the statistics of the data set is very important, and the domain knowledge also plays a vital role to make decisions for training a model. I also gained knowledge on various important factors like grid search and randomized search. I got to know that if we visually represent our data through different visualizations like (bar chart, histograms etc.) it gets easy to understand the dataset.

The difficulty I faced in the module was to understand the math's behind the algorithms and the equations which where there in the lectures like hypogenous function and the cost function. Some of the algorithms were a little challenging to understand like linear regression algorithm but I found my way through them.

If doing this project again I'll try to explore more advanced model like LSTM (Long Short-Term Memory) networks or Transformers, which are more effective for text data. I will focus more on precision and recall. For hypermeter tuning ill use some different techniques like Bayesian optimization.

This SMS Spam Detection project is remarkable, in aspects that set it apart from existing machine learning research in the field. These unique contributions and fresh ideas not improve the models performance. Also provide new insights into spam detection techniques. We go beyond methods of extracting text features. Our project incorporates a feature engineering strategy that combines TfidfVectorizer with approaches to capture the contextual meaning of messages. This approach enables our model to comprehend the nuances in text making it better at distinguishing spam, from messages.

X. REFERENCES

1. A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Master's thesis, Dept. of Comp. Sci., Univ. of Toronto, Toronto, ON, Canada, 2009.
2. T. Mitchell, "Machine Learning," New York, NY, USA: McGraw Hill, 1997.
3. I. H. Witten, E. Frank, and M. A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques," 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2011.
4. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in Proc. of the AAAI Workshop on Learning for Text Categorization, Madison, WI, USA, 1998, pp. 55-62.
5. Ashraf M. Kibriya, Eibe Frank, Bernhard phafringer and Geofery Holmes, "Multinomial Naïve Bayes For Text Catagorisation Revisited", University Of Waikato, Hamilton, New Zealand.