1. Introduction:

Corporations, businesses, and everyday life continually advance with technology, leveraging the latest algorithms, applications, and high-end devices to enhance convenience and efficiency. Amid these technological advancements, one constant remains pivotal—communication and the seamless flow of information.

From ancient smoke signals thousands of years ago to modern internet communication, our ability to connect and share information has evolved dramatically. However, as technology becomes more ubiquitous and sophisticated, individuals with nefarious intentions may seek to exploit it. This is evident in the prevalence of spam emails and scams, which have become a common phenomenon. With modern coding techniques, all it takes is a convincing persuasion and a single click on a malicious link to victimize an unsuspecting target. Fortunately, just as scammers utilize cutting-edge technology to their advantage, we can harness it to thwart their efforts. In this report, we endeavor to implement a spam email detection system using machine learning methods.

Section 2 briefly discusses the formulation of the problem; we'll go through the data we're analyzing. Section 3 describes the creation of the dataset and explains the choice of the ML-method. Section 4 then presents the ML methods used along with their results, and in section 5 we analyze our findings and conclusion.

The results are gone through in section 4 comparing the performance of the models and section 5 discusses about our conclusions.

2. Problem Formulation:

Our objective is to develop a machine learning algorithm capable of analyzing a substantial dataset of emails labeled as either spam (1) or non-spam (0). The algorithm's purpose is to identify patterns and similarities within the text of spam emails and leverage these patterns to detect future instances of spam. The dataset, sourced from Kaggle [1], serves as a rich source of data for our analysis. The labels in our dataset are denoted as "spam" and "ham," representing spam and non-spam, respectively. In this task, we employ supervised learning, as the algorithm learns from a labeled dataset.

3. Methods:

3.1. Data and preprocess

As mentioned, our dataset is ideal/large. To be exact, the dataset contains 5571 datapoints, which are useful to us.

```
      v1                                                 v2 Unnamed: 2  \
0    ham  Go until jurong point, crazy.. Available only ...        NaN
1    ham                      Ok lar... Joking wif u oni...        NaN
2   spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN
3    ham  U dun say so early hor... U c already then say...        NaN
4    ham  Nah I don't think he goes to usf, he lives aro...        NaN
...  ...                                                ...        ...
5567 spam  This is the 2nd time we have tried 2 contact u...        NaN
5568  ham              Will Ì_ b going to esplanade fr home?        NaN
5569  ham  Pity, * was in mood for that. So...any other s...        NaN
5570  ham  The guy did some bitching but I acted like i'd...        NaN
5571  ham                        Rofl. Its true to its name        NaN

     Unnamed: 3 Unnamed: 4
0           NaN        NaN
1           NaN        NaN
2           NaN        NaN
3           NaN        NaN
4           NaN        NaN
...         ...        ...
5567        NaN        NaN
5568        NaN        NaN
5569        NaN        NaN
5570        NaN        NaN
5571        NaN        NaN

[5572 rows x 5 columns]
```

Our dataset comprises 5571 data points, with key columns 'v1' and 'v2' representing the spam labels and email content, respectively. We conducted data cleaning to facilitate processing, renaming columns, labels, and limiting the dataset to Latin script for efficiency. Duplicate entries were also removed to ensure unbiased evaluation.

	Label	E-mail
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
dataframe = dataframe.drop_duplicates()
dataframe.shape
```

```
(5169, 2)
```

As we can see, the number of datapoints reduced.

3.2. Features and labels

The features in the problem are word frequencies in sentences are the chosen features. For each e-mail there was a label value, which told if the message was spam (1) or non-spam (0). The reason for this feature selection is obviously the fact that the messages are different, but spams will have some kind of similarity in terms of their content. This could be exaggerated lines like “click this link” or “YOU HAVE ONLY 24 HOURS”. The features were extracted with TfidfVectorizer()-function [2]. This technique transforms the emails into high-dimensional vectors where each dimension corresponds to a specific word or term and measures how important a word is to an email relative to all the words in a collection of emails.

3.3. ML models and loss functions

3.3.1. Logistic Regression

One of the chosen ML-methods for this problem is Logistic Regression [3] because it is ideal method for classification problems, which is exactly our case, because the e-mails are categorized to spams

and non-spams.
Logistic Regression can handle high dimensional data well, but it might suffer from overfitting if there are too many features compared to the number of training examples.

In terms of spam and non-spams, our dataset is rather unbalanced where non-spam emails significantly outnumber spam emails. Logistic Regression handles this by adjusting the class weights during training, giving more importance to under-represented classes.

The ideal loss function here is logistical loss function [4]. It provides a continuous and differentiable measure of the model's performance, making it suitable for optimization algorithms. Also, log loss penalizes confident and incorrect predictions more heavily, incentivizing calibrated probability estimates. Log loss can be interpreted as the logarithmic measure of the likelihood of the predicted probabilities aligning with the true labels.

3.3.2. Random Forest Classifier

The second chosen method is Random Forest Classifier (RFC) [5].
Like our first method, RFC is also ideal for classification methods. Due to our data going through various preprocess actions such as TF-IDF vectorizer, the data might be high dimensional. RFC handles this well because at each node in each decision tree, it only considers a random subset of features for splitting. This not only helps in managing high dimensionality but also provides an implicit form of feature selection.

Since RFC is an ensemble of multiple decision trees and it introduces randomness into the tree building process, it's more robust to overfitting compared to individual decision trees.

Like logistic regression, RFC handles unbalanced dataset well by providing balanced subsets of data for each decision tree to train on.

Decision trees are not differentiable; therefore, they don't use gradient descent, therefore they don't have a "loss" function in the typical sense. Rather, decision trees typically use "gini" impurity or "entropy" to select the best way to branch from each node in the tree given the training data. Gini is used to decide which features to split on at each node in the decision trees. It's a metric that quantifies the purity of a node or a set in the decision tree. In other words, it measures the probability of incorrectly classifying a randomly chosen element in the dataset if it were randomly labeled according to the class distribution in the dataset.

3.4. Data split and model validation

The decided split for data was 70% (training, 3618 datapoints), 15% (validation, 775 datapoints), 15% (test, 776 datapoints). This conclusion was reached after dataset size evaluation and the split's common occurrence in many research articles. 8:1:1 split would've been a good choice also. Each set then received a feature transformation using TfidfVectorizer()-function.

4. Results

Results for trained model	Accuracy	Precision	Error	F1 (spam)	F1 (non spam)
Training log reg	0.993	0.949	0.259	0.97	0.98
Training RFC	1	1	NaC	1	1

Validation log reg	0.963	0.908	1.302	0.85	0.98
Test log reg	0.975	0.929	0.88	0.91	0.99
Validation RFC	0.947	1	NaC	0.74	0.97
Test RFC	0.955	1	Nac	0.8	0.97

Performance of training data on trained models perform the best as is expected, because we check the performance of a data to the models that have been trained with the same data.

As we can see, both models perform well in terms of accuracy and precision with validation and test data as well. However, we can see, that RFC model doesn't do as well predicting spam emails in comparison to detecting non-spam emails. This can be due to our dataset being unbalanced favoring non-spam emails in terms of quantity. However, logistic regression seems to detect spams better.
In terms of accuracy, both models seem to perform similarly. The predictions seem to correspond the original data well, but we can see couple percent better performance from logistic regression compared to logistic RTC.
Training data with RFC method gives us the perfect accuracy and precision, which means that the method predicts emails perfectly.

As RFC doesn't have a loss function, there is no errors to be compared but with logistic regression we can again see the same pattern, where training data has the best value as it should and with the test set the error seems to be better than with validation.

When all of this is considered, we can assume logistic regression to be more ideal method for this problem. Although the precision is noticeably lower, the model seems to predict both spam and non-spam emails more consistently. As we can see, RFC struggles to predict the spam emails, which we can see not only in f1-score but also how it affects the accuracy making logistic regression come out on top.

5. Conclusion

In this report we studied a common everyday problem of spam emails. The used dataset contained thousands of emails labeled as spam or no spam. The set was preprocessed by removing duplicates and every single email was cleared from special characters and stop words and every word were in lower case. The data was split to training-, validation- and test sets with 70/15/15 split and as features we had the words used in emails.
The used methods were Logistic Regression and Random Forest Classification as both were ideal methods to use in classification problems. Both models performed well during our testing, but logistic regression came out top.
The end result was deemed to be serving the purpose of "spam email detection" rather well. The performances with validation- and test sets didn't differ too much from the training set, from which we can somewhat conclude that our model was trained well. However, the datasets could have been made more fair/less bias by using methods such as k-fold, so implementation could be improved.
The limitation of logistic regression is that it might struggle with multi-dimensional dataset, which requires parameter tuning. The limitation of RFC is, that it might be vulnerable to overfitting in case of noisy datasets. This could be improved with cross-validation.

Bibliography/References:

[1] https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset/code?resource=download

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html