SPAM CLASSIFICATION WITH MACHINE LEARNING AND TEXT MINING TECHNIQUES

1st Hilal Nur Tek
Gazi University, Computer Engineering
Ankara,Turkey
hilalnur.tek@gazi.edu.tr

Abstract— **With the developing technology and increasing internet access, the number and variety of data is increasing day by day. As a result of this growth, the development of communication channels has increased irresistibly. One of today's most used technologies, the telephone is one of the most important communication resources. The fact that it is very easy to reach and most communication channels are used over the phone also allows malicious people to manipulate this system. In this project, it is aimed to detect spam messages with various word embedding, text mining and machine learning techniques. The model trained with the Multinomial Naive Bayes algorithm after the count vector transformation from the compared Logistic regression, Naive Bayes and Random Forest algorithms was the most successful among the other models with an f1 score of 94.3%.**

Keywords— Text Mining, Text Classification, Spam Messages, Word Embedding, TF-IDF, Count Vector

I. INTRODUCTION

In our age, as the development and size of technology and internet increase, telephone and digital communication tools have started to take a more important place in our lives. Communication types such as letter and telegraph used in the past have been replaced by higher-level technologies suitable for daily life such as computers and telephones. The development of communication tools has started to be used very effectively by the business and marketing departments of many sectors such as banking and e-commerce. Users who are reached from various addresses such as phone numbers or e-mails can be encouraged to shop with various marketing techniques. If done correctly, communication channels that create positive effects for both the company and the user can also be used for bad actions. The use of these resources outside of their intended purpose can bring about a negative situation that may include many material or moral damages. Spam messages that will be processed in this study are an example of this misuse. This project deals with an artificial intelligence-based spam classification process developed to minimize the victimization that users may experience.

II. PURPOSE FOR PAPER

Due to the diversification of communication channels, text data is increasing day by day. Due to the ever-growing datasets, various tools and techniques are needed to process text documents and provide access to information. The most effective methods used in this context are text mining techniques. Text Mining, unstructured and irregular electronic text stacks; It is the process of obtaining previously unknown, potentially useful, structured and organized data. With the information obtained, relationships, hypotheses and trends that are not clearly seen in the analysed text sources are identified. Although Text Mining is considered as a part of data mining, it is different from conventional data mining. The main difference is that in Text Mining, patterns are extracted from natural language texts rather than event-based databases. Text Mining studies are data mining studies that accept text as a data source and aim to obtain structured data from text. Within the scope of Text Mining studies; information retrieval, lexical analysis, word frequency distribution, pattern recognition, tagging, information extraction, data mining and visualization methods are used. Text Mining studies are often discussed together with Natural Language Processing (NLP) studies, which is another field of study in the text-based literature. Natural language processing studies mostly cover linguistics-based studies under artificial intelligence.

Text Mining studies, on the other hand, aim to reach statistical results through text. During Text Mining studies, feature extraction is often done using natural language processing. Today, the amount of unstructured and disordered electronic text stacks is increasing at a geometric rate, and it becomes more and more important to obtain useful, structured and organized information from these data.

III. LITERATURE REVIEW

Since spam messages are quite disturbing today, there are many studies in the literature on the solution of this problem with various text processing and machine learning techniques. In [1] Tarika Verma et al. aim to detect e-mail spams via text mining and machine learning techniques. They use SVM, Naïve Bayes, Decision Tree, Linear Regression, Association Rule-based algorithms. In [2] Linda Huang, Julia Jia et al. (2018) developed a method for separating individual or company spam emails using the Naive Bayes algorithm. In [3], Nida Mirza et al. (2017) data mining and word count encode method obtained the best result in Naive Bayes algorithm. In [4], Maria Habib et al. (2018) proposed a system that uses the SMOTE technique to stabilize the unbalanced dataset and uses the genetic algorithm to detect a spam. In [5], R. Shams et al. (2013) proposed a system that processes the data processed by data mining with Random Forest, Naive Bayes, SVM, BAGGING, AdaBoostand algorithms. Among the algorithms that S. Saha et al. (2019) [6] compared, the Naive Bayes algorithm provided the best result. In [7], Mirza, N. et al. provided a new hybrid feature selection system to classify spam messages. In [8], Sunday Olusanya Olatunji (2017) proposed a SVM based e-mail spam detection system.

IV. METHODOLOGY

First Look

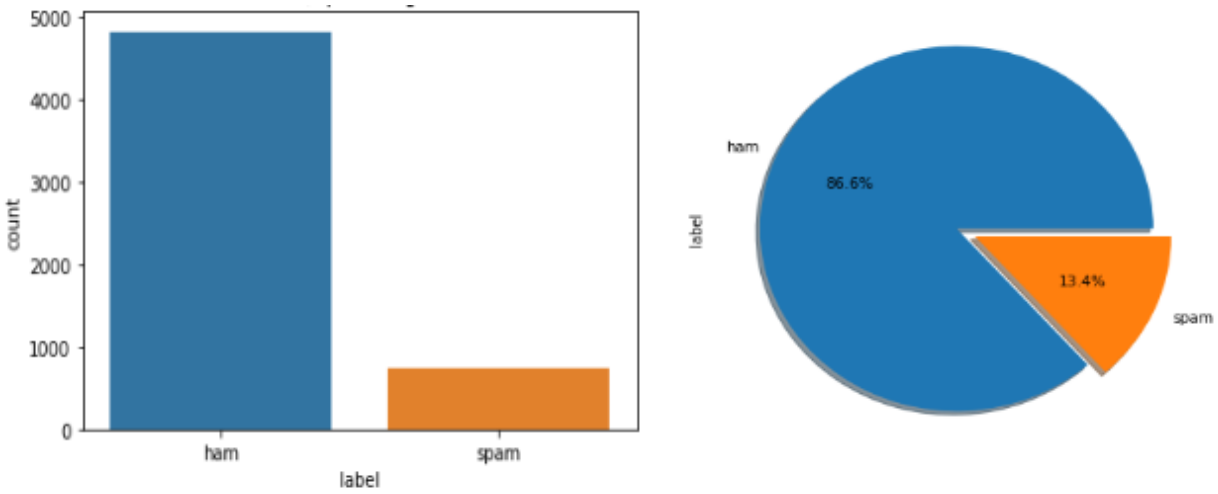
In machine learning projects, it is essential to first understand the data structure in order to properly manage and move the project forward. In this context, it is necessary to understand the dataset thoroughly before proceeding to the cleaning and training phase. When we look at the structure of the data set used, it is observed that it consists of two columns named message and label of object type and 5574 values (Table 4.1.). The message column represents the content of the SMS, and the label column represents whether the SMS is spam or not. In addition, as can be seen in the table below, the number of messages in the data set, unique values, the most common message and label values were also examined.

```
RangeIndex: 5574 entries, 0 to 5573
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   message  5574 non-null   object
 1   label    5574 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

	message	label
count	5574	5574
unique	5171	2
top	Sorry, I'll call later	ham
freq	30	4827

(Table 4.1.) Structure of the dataset

When the spam and normal message distribution in the data set is examined, it has been observed that the spam messages are much less than the normal messages (Table 4.2) and the data set is unbalanced.

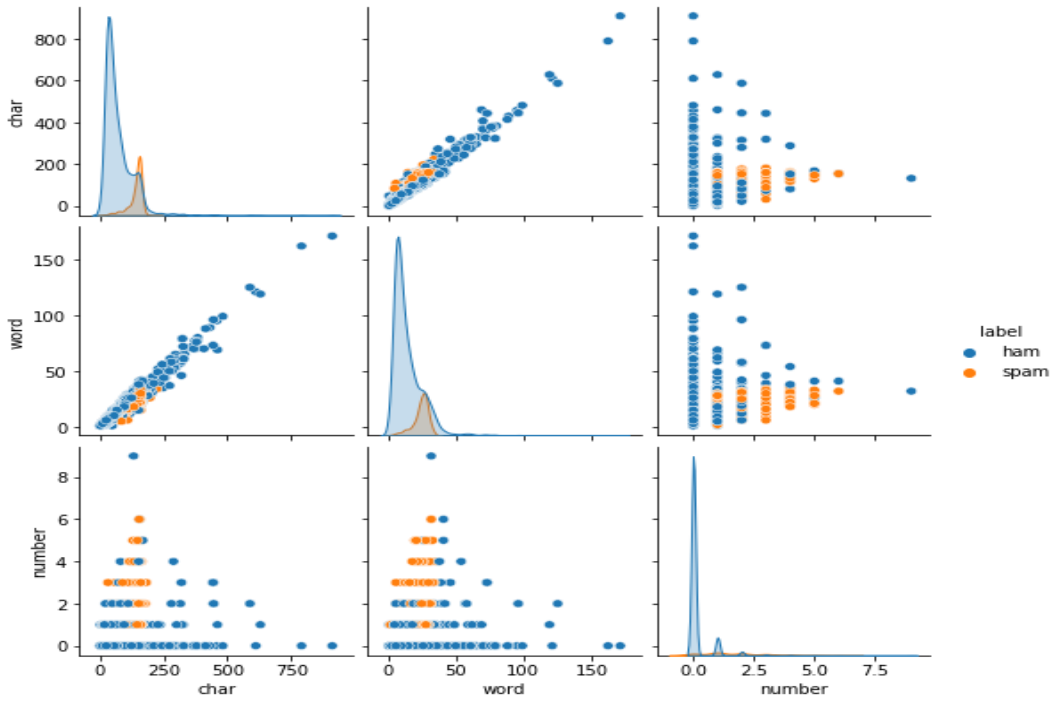


(Table 4.2.) Distribution of ham and spam messages

After the structure of the data set was examined and visualized, mathematical operations and simple feature extraction phase were started. At this stage, it was aimed to extract numerical expressions from the text and the number of letters/characters, word counts and numbers (Table 5.3. & Table 5.4.) in the texts were examined. The goal is to generate a numerical value to represent each of the observation units.

	text	label	char	word	number
0	Go until jurong point, crazy.. Available only ...	ham	111	20	0
1	Ok lar... Joking wif u oni...	ham	29	6	0
2	Free entry in 2 a wkly comp to win FA Cup fina...	spam	155	28	2
3	U dun say so early hor... U c already then say...	ham	49	11	0
4	Nah I don't think he goes to usf, he lives aro...	ham	61	13	0

(Table 4.3.) Counts of characters, words and numbers for each text



(Table 4.4.) Visualization of character, word and number counts for each text

Pre-Processing

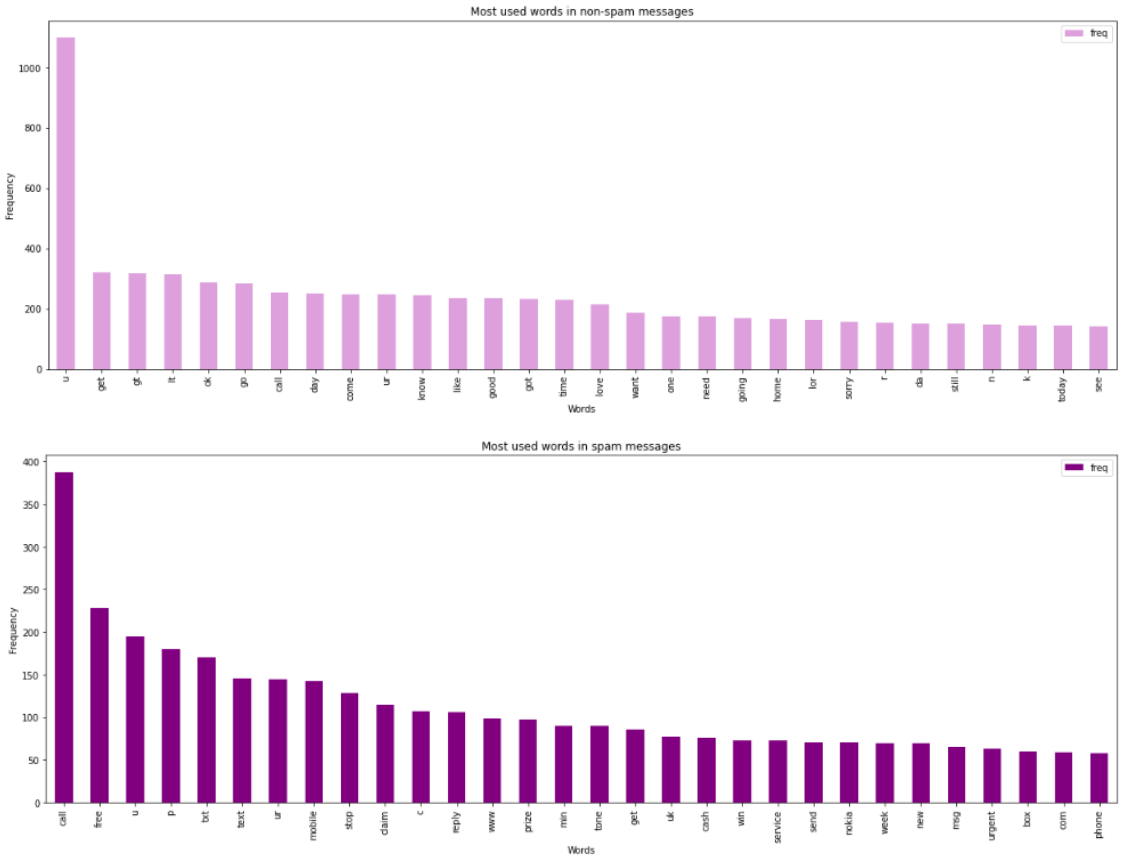
Before training text data, it must go through certain pre-processing steps so that model performance is not adversely affected. In order for the machine learning model to be created to make sense of the unstructured data, it is essential to make the data workable. In order for the data to meet the workability criterion;

- **Separation of Special Characters**: Use of letters found in the Latin alphabet only.
- **Upper-Small Character Conversion**: Expressing all characters with lowercase letters so that the upper and lower case characters of the same letter are not perceived differently by the model.
- **Stop Words:** Removal of expressions such as “I, you, the, at” in order not to impair the accuracy of the model.
- **Deletion of Infrequently Found Expressions**: Deletion of rarely used expressions in the text as they do not affect the classification.

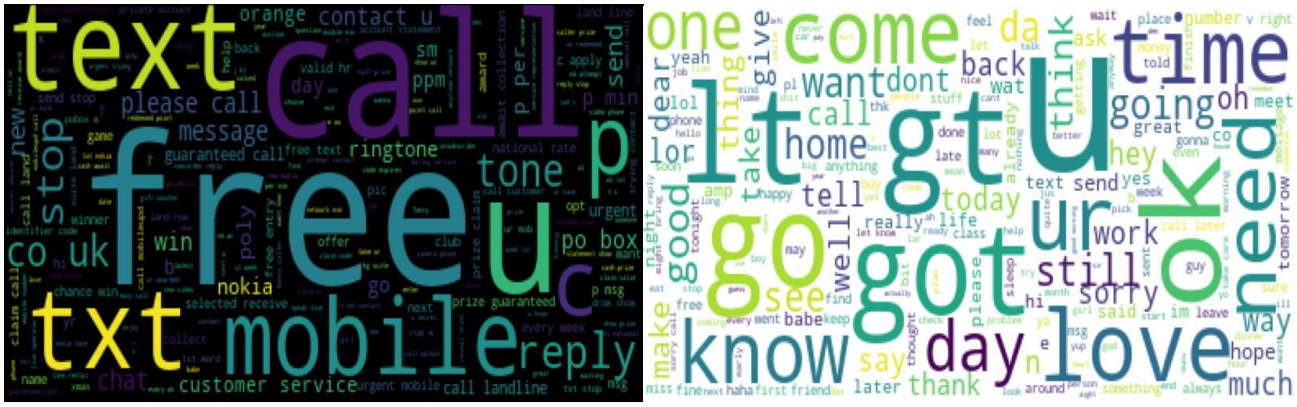
Normalization

Normalization helps to group words that have the same meaning but different shapes. Without normalization, words will be treated as different words even if they have the same root structure. In this study, "Lemmatization" technique, which is one of the popular normalization techniques, was used. In this way, even if the words with the same root have different conjugations, they are not perceived differently by the model because they are reduced to the root form.

The visualization operations performed after the pre-processing steps provide an overview of the structure of the text. In the tables (Table 5.5. & Table 5.6) given below, the frequency of words in spam and non-spam messages is visualized by bar plot and word cloud techniques.



(Table 4.5.) Most used words in spam and non-spam messages visualized with bar plot



(Table 4.6.) Most used words in spam and non-spam messages visualized with word cloud

Word Embedding Techniques

Verbal expressions need to be made meaningful for machine learning or deep learning algorithms. For this reason, words must be expressed numerically. Algorithms such as Count Vector, TF-IDF, as word embedding techniques used to solve such problems, allow words to be expressed mathematically. One of these techniques (in some cases several) is preferred and used according to the status, size and purpose of processing the data.

One of the most basic techniques used to express data numerically is One Hot Encoding technique [9-10]. In this method, a vector is created in the size of the total number of verbal expressions. The value of vectors is assigned such that the value of each verbal expression belonging to its index is 1

and the others are 0. It is generally used in situations where there is not much verbal data diversity and there is no need to represent the semantic and statistical relationships between the data.

TF-IDF is a statistical measure used to determine the mathematical significance of words in documents [11-12]. The vectorization process is similar to One Hot Encoding. Alternatively, the value corresponding to the verbal expression is assigned a TF-IDF value instead of 1. The TF-IDF value is obtained by multiplying the TF and IDF values. In the simplest terms, term frequency is the ratio of the number of target terms in the document to the total number of terms in the document. The IDF value is the logarithm of the ratio of the total number of documents to the number of documents in which the target term occurs. At this stage, it does not matter how many times the term appears in the document. It is sufficient to determine whether it has passed or not. In this study, 4 different word embedding techniques were studied in namely Count Vectors and TF-IDF Vectors (words, characters, n-grams).

At this stage, the text data is then encoded and converted into numeric data. The converted data is first exposed to count vector (Table 5.7.), then character, word and n-gram based TF-IDF (Table 5.8.) word embedding techniques. Then the data used to train the model is first divided into two as X and y. X represents the values we want to make sense of (text data in this project) and y represents target data (spam or raw in this project). Then the data is divided in the desired ratio to be trained and tested. After the model is trained with train sets, it is checked with test sets and performance values are calculated.

Count Vectors

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train)
```

```
CountVectorizer()
```

```
x_train_count = vectorizer.transform(X_train)
x_test_count = vectorizer.transform(X_test)
```

```
vectorizer.get_feature_names()[:5]
```

```
['aa', 'aah', 'aaooooright', 'aathi', 'abdomen']
```

```
x_train_count.toarray()
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

(Table 4.7.) Transforming data with the Count Vector technique

Word Level TF-IDF

```
tf_idf_word_vectorizer = TfidfVectorizer()
tf_idf_word_vectorizer.fit(X_train)
x_train_tf_idf_word = tf_idf_word_vectorizer.transform(X_train)
x_test_tf_idf_word = tf_idf_word_vectorizer.transform(X_test)
```

```
tf_idf_word_vectorizer.get_feature_names()[:5]
```

```
['aa', 'aah', 'aaooooright', 'aathi', 'abdomen']
```

Ngram Level TF-IDF

```
tf_idf_ngram_vectorizer = TfidfVectorizer(ngram_range = (2,3))
tf_idf_ngram_vectorizer.fit(X_train)
x_train_tf_idf_ngarm = tf_idf_ngram_vectorizer.transform(X_train)
x_test_tf_idf_ngram = tf_idf_ngram_vectorizer.transform(X_test)
```

```
tf_idf_ngram_vectorizer.get_feature_names()[:5]
```

```
['aa exhaust',
 'aa exhaust hanging',
 'aah cuddle',
 'aah cuddle would',
 'aaooooright work']
```

Characters level TF-IDF

```
tf_idf_chars_vectorizer = TfidfVectorizer(analyzer = "char", ngram_range = (2,3))
tf_idf_chars_vectorizer.fit(X_train)
x_train_tf_idf_chars = tf_idf_chars_vectorizer.transform(X_train)
x_test_tf_idf_chars = tf_idf_chars_vectorizer.transform(X_test)
```

```
tf_idf_chars_vectorizer.get_feature_names()[:5]
```

```
[' a', ' aa', ' ab', ' ac', ' ad']
```

(Table 4.8.) Transforming data with the different kind of TF-IDF techniques

Model Training

During the model creation phase, it was decided to train the data with 3 classification models. These are Logistic Regression [13], Naive Bayes [14] and Random Forests Classifier [15]. Before proceeding to the result evaluation, it is necessary to know how these algorithms work.

Logistic regression is a classification algorithm and it uses the Sigmoid (Logistics) Function (Figure 5.1) to classify. The sigmoid function is simply the function used to compress our data between 0 and 1.

$$g(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-(\theta^T x)}}$$

(Figure 5.1.) Sigmoid Function Formula

$$Cost(h_\theta(x),y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

(Figure 5.2.) The cost function of the Logistic Regression

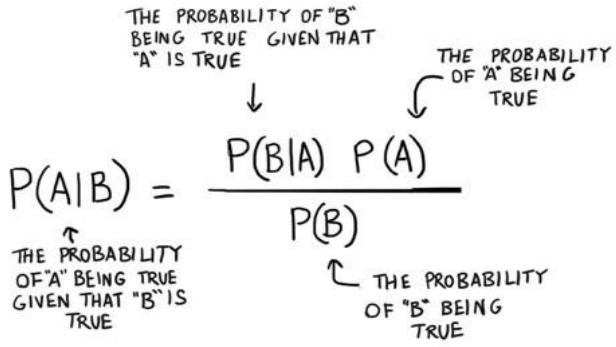
Advantages:

- Logistic regression is easy to implement and interpret.
- It performs quite well if the dataset is linearly separable.
- It is less prone to overfitting but can be over fitted on large datasets.

Disadvantages:

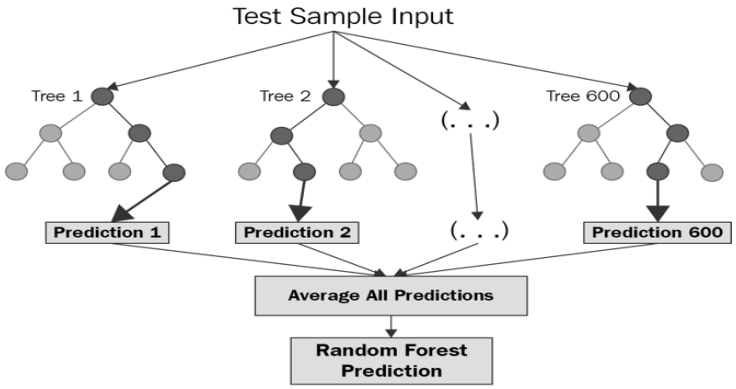
- If the number of observations is less than the number of features, Logistic Regression should not be used, otherwise over fit may occur.
  For logistic regression to discriminate, the data set must be linearly separable.

Naive Bayes classifier is based on Bayes theorem (Figure 5.3). It is a lazy learning algorithm, but it can also work on unstable datasets. For this reason, it is a frequently used algorithm in spam detection studies. The way the algorithm works is it calculates the probability of each state for an element and classifies it according to the highest probability value. It can produce very successful works with a little training data. If a value in the test set has an unobservable value in the training set, it returns 0 as a probability value, so it cannot predict. This condition is commonly known as Zero Frequency. Corrective techniques can be used to resolve this situation. One of the simplest correction techniques is known as Laplace estimation.



(Figure 5.3.) Bayes Formula

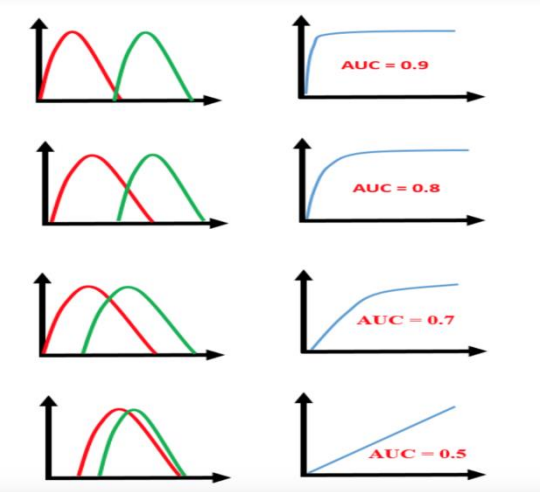
Random forest algorithm is one of the supervised classification algorithms. It is used in both regression and classification problems. The algorithm aims to increase the classification value during the classification process by producing more than one decision tree. Random forest algorithm is the process of choosing the highest score among many decision trees that work independently of each other. As the number of trees increases, our rate of obtaining a precise result increases. The main difference between the decision tree algorithm and the random forest algorithm is that the process of finding the root node and splitting the nodes is random. The random forest algorithm reduces the problem of over-learning if you have enough trees. It requires little data preparation.



(Figure 5.4.) Random Forest Algorithm

V. FINDINGS

Although the Accuracy value is frequently used in performance evaluations, it may not give accurate results, especially for an unevenly distributed data set. While considering the Accuracy value, different metrics calculated with Confusion Matrix values, which is one of the methods frequently used to define performance in classification problems, are also included in the performance table. The working logic of the Confusion Matrix is explained in the Fıgure 5.1. below.



(Figure 5.2.) AUC score explanation

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

(Figure 5.1.) Confusion matrix

Accuracy = (TP + TN) / (TP + FP + TN + FN)
Precision = TP / (TP + FP)
Recall = TP / (TP + FN)
F1-score = 2 * Precision * Recall / (Precision + Recall)

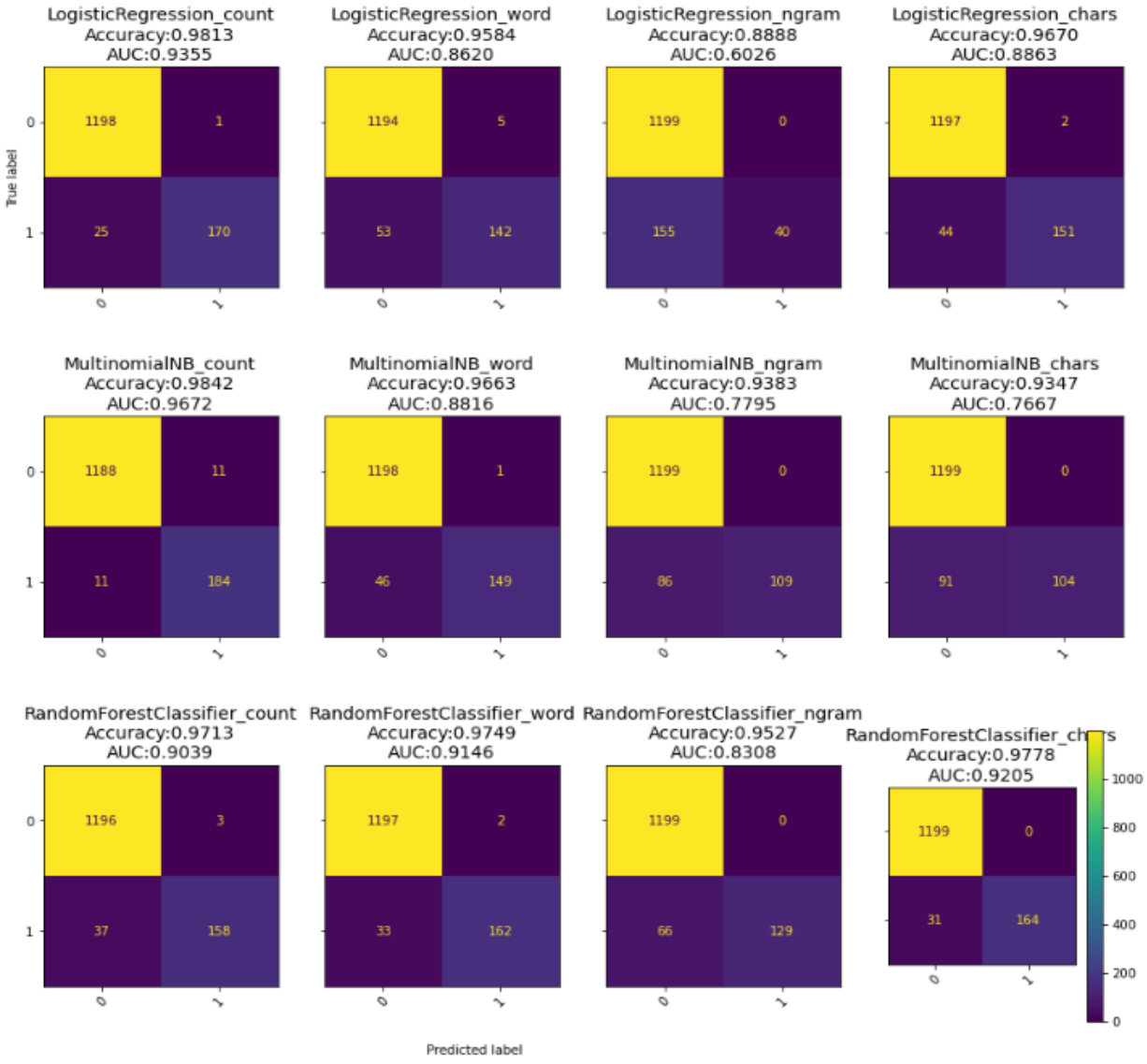
Another performance measure, AUC, represents the measure or degree of separability of parameters. In Figure 5.2. , it can be observed to what extent the classes in which the model results will be predicted differ. The working logic of the AUC metric is shown in the image below. The curves represented in green and red are the raw and spam values of the project. An increase in AUC indicates that the model can successfully distinguish these two classes.

	model_name	training_set_score	test_set_score	precision	recall	f1_score	AUC
4	MultinomialNB_count	0.990909	0.984218	0.943590	0.943590	0.943590	0.967208
0	LogisticRegression_count	0.994976	0.981349	0.994152	0.871795	0.928962	0.935480
11	RandomForestClassifier_chars	1.000000	0.977762	1.000000	0.841026	0.913649	0.920513
9	RandomForestClassifier_word	1.000000	0.974892	0.987805	0.830769	0.902507	0.914551
8	RandomForestClassifier_count	1.000000	0.971306	0.981366	0.810256	0.887640	0.903877
3	LogisticRegression_chars	0.975120	0.967001	0.986928	0.774359	0.867816	0.886345
5	MultinomialNB_word	0.977273	0.966284	0.993333	0.764103	0.863768	0.881634
1	LogisticRegression_word	0.970096	0.958393	0.965986	0.728205	0.830409	0.862017
10	RandomForestClassifier_ngram	0.999761	0.952654	1.000000	0.661538	0.796296	0.830769
6	MultinomialNB_ngram	0.972727	0.938307	1.000000	0.558974	0.717105	0.779487
7	MultinomialNB_chars	0.945694	0.934720	1.000000	0.533333	0.695652	0.766667
2	LogisticRegression_ngram	0.897368	0.888809	1.000000	0.205128	0.340426	0.602564

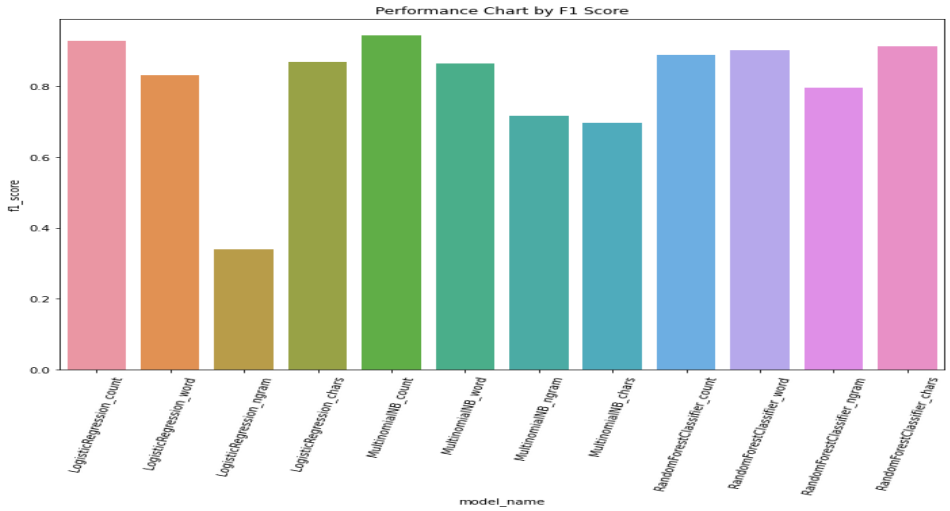
(Table 5.3.) Performance table according to success metrics

According to Table 5.3, it was seen that the most successful machine learning model was the Naive Bayes classifier, and the most successful word embedding method was the counting vector. However, according to the second and last data (LogisticRegression_count, LogisticRegression_ngram), data processed with different encoding methods can give different outputs even if they are trained by the same algorithm. This result shows us how important word embedding techniques are in text processing projects. According to this study, it can be said that the most successful coding method is the counting vector. n addition, it has been observed that the algorithm that gives the most stable results, even if it is not the most successful, is the Random Forest Classifier.

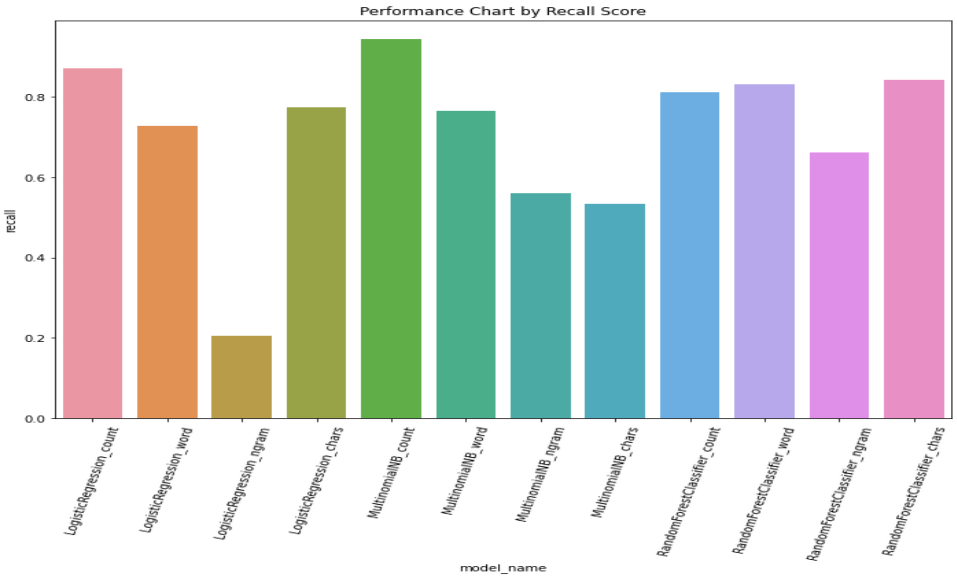
When the performance metrics are compared, it is seen that there are big differences in the success metrics of the algorithms in the last row. Large differences between success metrics indicate overfitting of the model. It is undesirable for the model to be overfitting because it means that our model does not learn from the given data, but memorizes them.



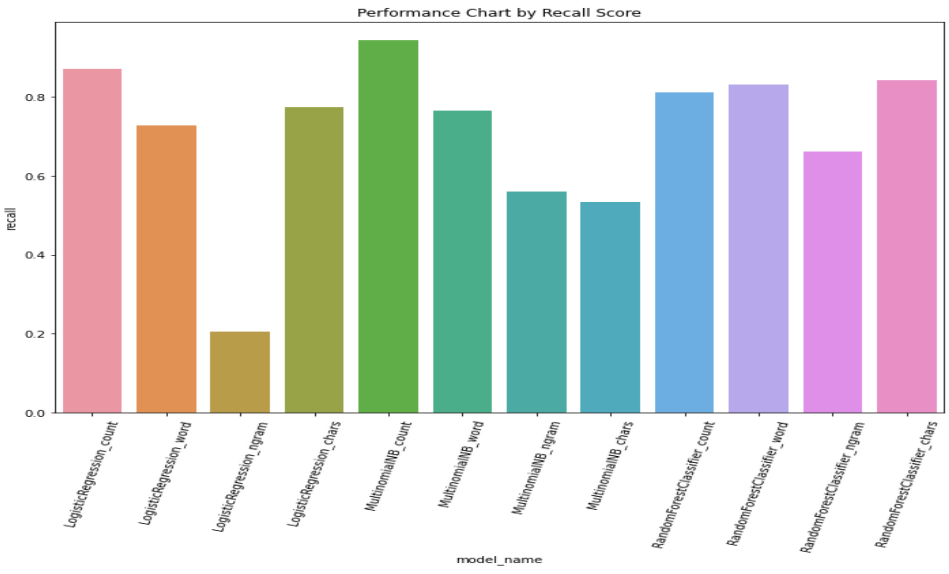
(Table 5.4.) Confusion matrixes of the trained models



(Table 5.5.) Performance chart by f1 score



(Table 5.6.) Performance chart by precision score



(Table 5.7.) Performance chart by recall score

VI. CONCLUSION

The project aims to prevent spam messages in order to minimize the victimization of users. In this direction, text mining and machine learning algorithms, which are widely used in the literature, were used. After examining the dataset with an overview, it has been subjected to certain pre-processing processes. After these processes, the text values are converted to numerical values and separated as train/test, and the model is ready for training. Three classification algorithms, which are frequently found in the literature, were preferred to train the model. These are Logistic Regression, Naive Bayes and Random Forest Classifier. According to the performance metrics obtained, it has been observed that the best performance among the three algorithms given, with 94.3%, belongs to Multinomial Naive Bayes, which was encoded with the count vector method. Looking at all the results obtained, it has been observed that word embedding techniques have an important place in the operations performed on the text and significantly affect the model performance. The aim of this study is to provide a general framework for future studies.

VII. REFERENCES

[1] Verma, Tarika & Gill, Nasib. (2020). Email Spams via Text Mining using Machine Learning Techniques. 9. 2535-2539. 10.35940/ijitee.D1915.029420.

[2] W. Peng, L. Huang, J. Jia and E. Ingram, "Enhancing the Naive Bayes Spam Filter Through Intelligent Text Modification Detection," 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, 2018, pp. 849-854. doi: 10.1109/TrustCom/BigDataSE.2018.00122

[3] Mirza, N., Patil, B., Mirza, T., & Auti, R. (2017). “Evaluating efficiency of classifier for email spam detector using hybrid feature selection approaches”. 2017 International Conference on Intelligent Computing and Control Systems (ICICCS). doi:10.1109/iccons.2017.8250561

[4] Habib, M., et al. (2018). “Automatic Email Spam Detection using Genetic Programming with SMOTE”. Fifth HCT Information Technology Trends (ITT). doi:10.1109/ctit.2018.8649534

[5] R. Shams and R. E. Mercer, “Classifying spam emails using text and readability features,” Proc. - IEEE Int. Conf. Data Mining, ICDM, pp. 657–666, 2013.

[6] Saha S., DasGupta S., Das S.K. (2019), “Spam Mail Detection Using Data Mining: A Comparative Analysis”. In: Satapathy S., Bhateja V., Das S. (eds) Smart Intelligent Computing and Applications. Smart Innovation, Systems and Technologies, vol 104. Springer, Singapore. Doi: 10.1007/978-981-13-1921-1_56

[7] Mirza, N., Patil, B., Mirza, T., & Auti, R. (2017). “Evaluating efficiency of classifier for email spam detector using hybrid feature selection approaches”. 2017 International Conference on Intelligent Computing and Control Systems (ICICCS). doi:10.1109/iccons.2017.8250561

[8] Olatunji, S. O. (2017). “Improved email spam detection model based on support vector machines” Neural Computing and Applications. doi:10.1007/s00521-017-3100-y

[9] Seger, Cedric. "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing." (2018).

[10] Stevens, S. S. (1946). “On the Theory of Scales of Measurement”. Science, New Series, 103.2684,677–680.

[11] Yun-tao, Z., Ling, G., & Yong-cheng, W. (2005). An improved TF-IDF approach for text classification. Journal of Zhejiang University-Science A, 6(1), 49-55.

[12] Aizawa Akiko, (2003). “An information-theoretic perspective of tf–idf measures”. Information Processing and Management. 39 (1), 45–65. doi:10.1016/S0306–4573(02)00021–3

[13] Wright, R. E. (1995). Logistic regression.

[14] Webb, G. I., Keogh, E., & Miikkulainen, R. (2010). Naïve Bayes. Encyclopedia of machine learning, 15, 713-714.

[15] Pal, M. (2005). Random forest classifier for remote sensing classification. International journal of remote sensing, 26(1), 217-222.