Suspicious Chat Detection with Machine Learning

Cen-439 Web Application Security Project

Computer Engineers Students
S Gökcen Ozden
Akif Artun

Abstract: With the increase in the use of electronic devices, it has become easier for people of all ages to communicate with each other through these devices. Especially with the widespread use of Internet and the popularity of social media channels, communication is now much faster than before. However, this widespread use also paves the way for disadvantages and new abuses. It is one of them that malicious correspondence and suspicious behavior leave innocent users and especially children vulnerable. With this study, we aimed to protect young brains from evil by controlling 'suspicious chat detection' with the help of machine learning. Thus, a written slang will manage to be filtered before it reaches the other side. Our dataset contains a total of 24,783 tweets.We have classified our dataset by following natural language processing procedures. In addition to each tweet, there are also certain features of these tweets. Tensorflow, Random Forest and Naive Bayes Multinominal approaches were discussed in the processing of these texts, which are all in English, and the most successful approach was determined as Tensorflow.

Keywords: natural language Processing; suspicious text detection;English language processing; machine learning; text classification; suspicious corpora; feature extraction

———————————————————————————————————

1.Introduction

Due to the effortless access of the Internet, world wide web, blogs, social media, discussion forums, and online platforms via digital gadgets have been producing a massive volume of digital text contents in recent years. It is observed that all the contents are not genuine or authentic; instead, some contents are faked, fabricated, forged, or even suspicious.

The data we collect from a public data source such as Twitter is the most natural data obtained because it comes directly from a communication channel. As a matter of fact, since this data contains various spelling errors, they should first be cleaned and pre-processed. The data allocated to the tokens are divided into 'hate speech', 'offensive_language' and neither tags and is numbered according to its severity.[1]

The model, which we have trained with a clean dataset, will naturally be met with non-clean CORPUS. For this, we got help from the Pyspell python library. It directly contributes to the classification of the text by correcting the word said statistically. And this enables the system that we have created more effectively than expected to work.

We have also performed a comparative analysis of these machine learning models utilizing our collected datasets. The key contributions of our work are illustrated in the following:

* Develop a corpus containing 24000 text documents labelled as suspicious or non-suspicious.

*Design a classifier model to classify English text documents into suspicious or non-suspicious categories on developed corpus by exploring different feature combination.

*Compare the performance of the proposed classifier with various machine learning techniques as well as the existing method.

*Analyze the performance of the proposed classifier on different distributions of the developed dataset.

2.Related Work

A machine learning-based system developed to detect promotion of terrorism by analyzing the contents of a text. Iskandar et al. [2] have collected data from Facebook, Twitter, and numerous micro-blogging sites to train the model. By performing a critical analysis of different algorithms, they showed that Naïve Bayes is best suited for their work as it deals with probabilities [3].

3.Dataset : Suspicious and non-suspicious Tweets:[4]

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya 

DataFrame Structure:

The first table shows an example of the first five rows of the DataFrame, including the following columns:

Count: Number of people who labelled the tweet.

Hate_speech: Number of labellers who classified the tweet as hate speech.

Offensive_language: Number of labellers who classified the tweet as offensive language.

Neither: Number of labellers who considered that the tweet does not fall into any of the previous categories.

Class: Final category assigned to the tweet, where 0 can represent 'hate_speech', 1 can represent 'offensive_language', and 2 can represent 'neither'.

Tweet: The textual content of the tweet.

```
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   count               24783 non-null  int64
 1   hate_speech         24783 non-null  int64
 2   offensive_language  24783 non-null  int64
 3   neither             24783 non-null  int64
 4   class               24783 non-null  int64
 5   tweet               24783 non-null  object
dtypes: int64(5), object(1)
memory usage: 1.1+ MB
Veri seti bilgileri:
 None
Rastgele örnek kayıtlar:
        count  hate_speech  offensive_language  neither  class  \
11537      3            0                   3        0      1
17897      9            1                   8        0      1
12077      3            1                   2        0      1
21442      3            0                   1        2      2
10276      3            0                   3        0      1

                                                   tweet
11537  If anything I've ever said has offended you, t...
17897  RT @Tylar____: @1stBlocJeremiah lol man bitch ...
12077  Joe Budden always flexin on average twitter ni...
21442                    That'd been a ghetto little meal
10276                               I eats the pussy.
Sınıf dağılımı:
 class
1    19190
2     4163
0     1430
Name: count, dtype: int64
```

Main Notebook Content

	count	hate_speech	offensive_language	neither
[illegible]	[illegible]000000	24783.000000	24783.000000	24783.000000
mean	3.243473	0.280515	2.413711	0.549247
std	0.883060	0.631851	1.399459	1.113299
min	3.000000	0.000000	0.000000	0.000000
25%	3.000000	0.000000	2.000000	0.000000
50%	3.000000	0.000000	3.000000	0.000000
75%	3.000000	0.000000	3.000000	0.000000
max	9.000000	7.000000	9.000000	9.000000

	class
count	24783.000000
mean	1.110277
std	0.462089
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	2.000000

Statistical Description:

The second table is a statistical summary of the DataFrame with the following metrics for each numeric column:

Count: Total records for each column.

Mean: Average of values.

Std (standard deviation): Measures the amount of variation or dispersion of values.

Min: Minimum value found. 25% (first quartile): Below this value are the 25% lowest values. 50% (median): Half of the values are lower than this value. 75% (third quartile): Below this value are the 75% lowest values.

Max: Maximum value found.

Statistical Data Analysis: * count (count): 24,783 tweets were labelled, which indicates a relatively large data set.

Hate_speech (mean): On average, 0.28 tweet labellers consider a tweet as hate speech, which suggests that most tweets are not considered as such.

Offensive_language (mean): On average, 2,41 labellers per tweet consider a tweet as offensive language, indicating that it is more common to find offensive language than hate speech.

Neither (mean): On average, 0.55 labellers per tweet found neither hate speech nor offensive language in the tweets.

Class (mean): The average close to 1.11 for the final category suggests that most tweets are being classified as offensive language (1), with some classified as neither offensive nor hate speech (2). *

4.PreProcessing

```
def clean_text(text):
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    return text
```

We apply cleaning functions to the text of tweets to remove unwanted elements such as URLs, HTML tags and special characters. In addition, we convert all text to lowercase to standardise the input before tokenisation and vectorisation. This step is essential to prepare the data for effective modelling.

```
df['clean_text'] = df['tweet'].apply(clean_text)
df['tokens'] = df['clean_text'].apply(word_tokenize)
```

5.Process

5.1 Creating a TF-IDF Matrix Using “TfidfVectorizer”

First, create the TF-IDF matrix using TfidfVectorizer and convert it to a Dense (dense) format:

```
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(df['clean_text'])
X_tfidf_dense = X_tfidf.toarray()
```

Using SMOTE, we re-sample the TF-IDF matrix and class labels.

```
smote = SMOTE()
X_smote, y_smote = smote.fit_resample(X_tfidf_dense, df['class'])
```

We define and compile the model

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_dim=X_smote.shape[1]),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

We separate the training set to 80% train and 20% test and sign our 42 random state.

5.2 RANDOM FORREST

```python
# Random Forest Modeli
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_y_pred = rf_model.predict(X_test)
print("Random Forest Classification Report:\n", classification_report(y_test, rf_y_pred, target_names=['Class 0', 'Class 1', 'Class 2']))
```

5.3 NAIVE BAYES MULTINOMINALNB

```python
# Naive Bayes Modeli
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_y_pred = nb_model.predict(X_test)
print("Naive Bayes Classification Report:\n", classification_report(y_test, nb_y_pred, target_names=['Class 0', 'Class 1', 'Class 2']))
```

6. REPORTS of Models

In this study, we especially focussed on our Tensorflow model and the results we received did not surprise us.

TensorFlow Model Classification Report:

	precision	recall	f1-score	support
Class 0	0.98	1.00	0.99	3849
Class 1	0.99	0.96	0.98	3794
Class 2	0.98	0.99	0.99	3871
accuracy			0.98	11514
macro avg	0.98	0.98	0.98	11514
weighted avg	0.98	0.98	0.98	11514

Random Forest Classification Report:

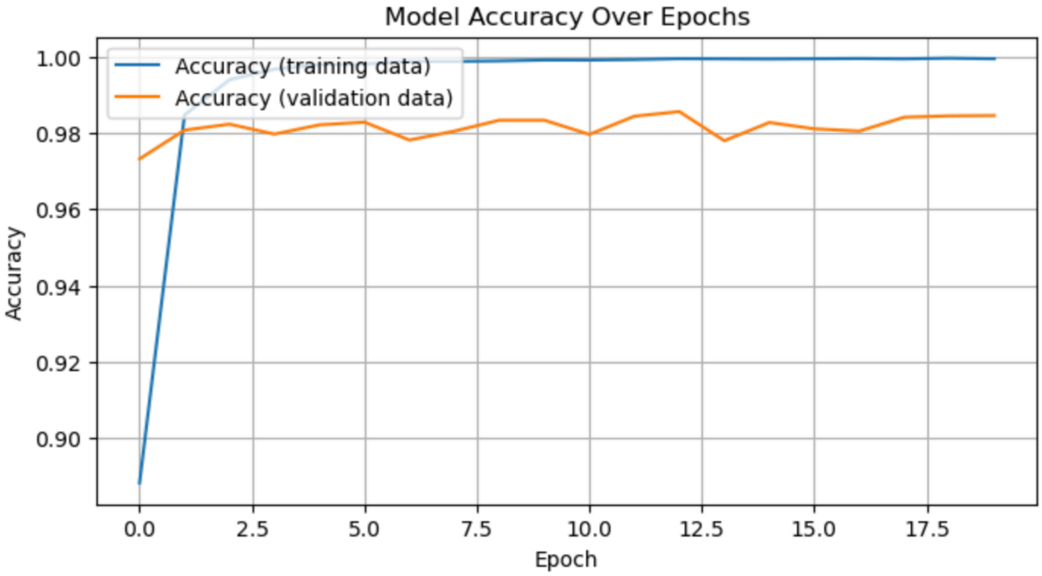
	precision	recall	f1-score	support
Class 0	0.98	0.99	0.99	3849
Class 1	0.97	0.95	0.96	3794
Class 2	0.97	0.98	0.97	3871
accuracy			0.97	11514
macro avg	0.97	0.97	0.97	11514
weighted avg	0.97	0.97	0.97	11514

Naive Bayes Classification Report:

	precision	recall	f1-score	support
Class 0	0.88	0.98	0.92	3849
Class 1	0.95	0.83	0.88	3794
Class 2	0.95	0.97	0.96	3871
accuracy			0.92	11514
macro avg	0.93	0.92	0.92	11514
weighted avg	0.93	0.92	0.92	11514

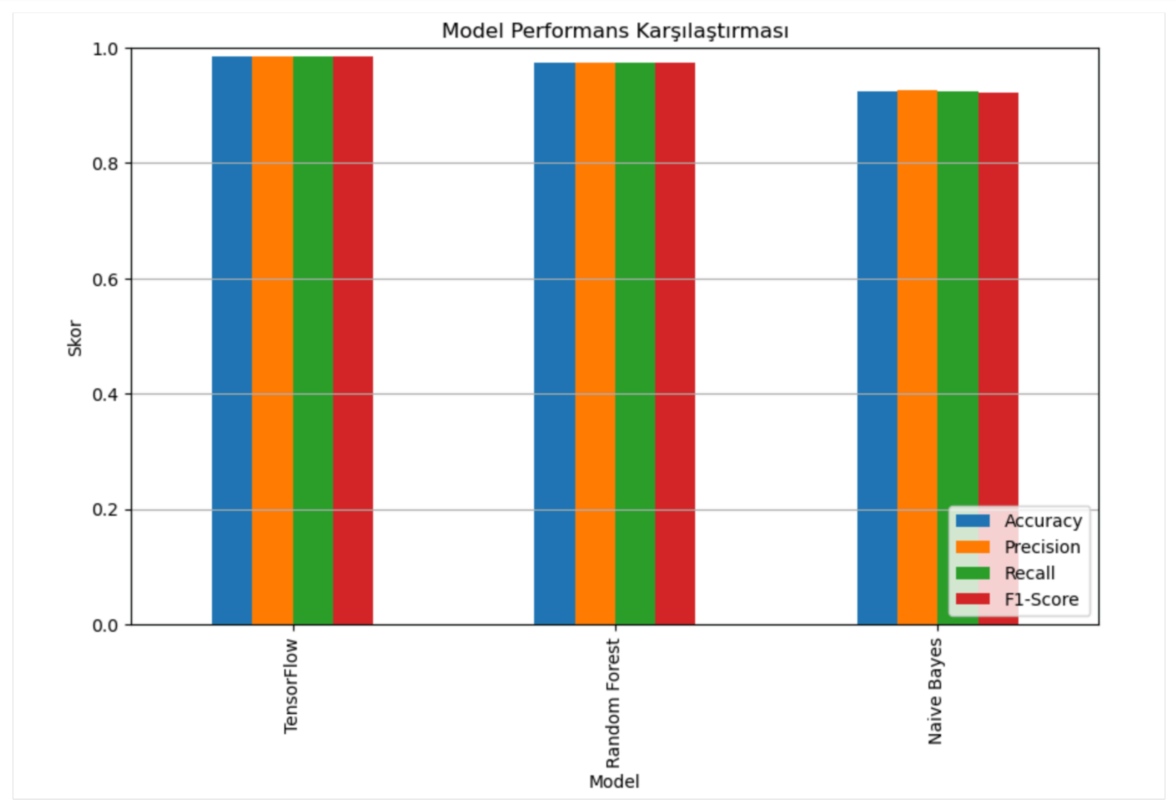
Tensorflow Performance

```
Accuracy: 0.9845405593190898
```

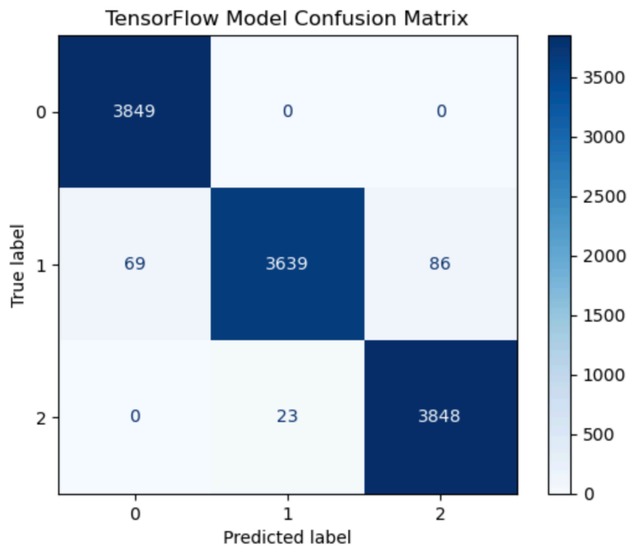


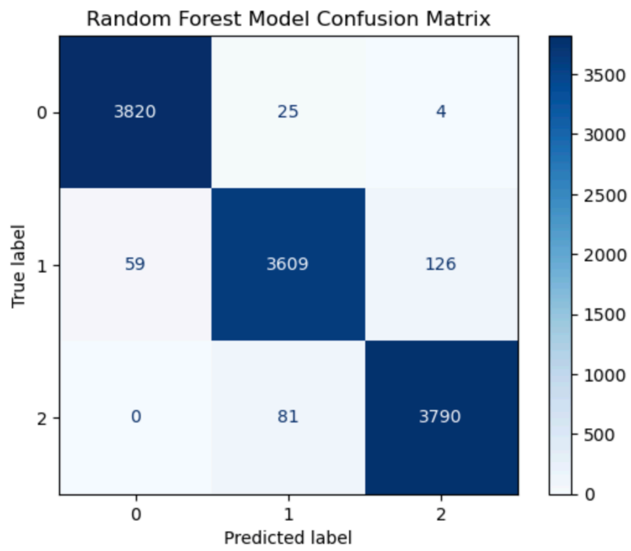
	Model	Accuracy	Precision	Recall	F1-Score
0	TensorFlow	0.984541	0.984694	0.984541	0.984469
1	Random Forest	0.974379	0.974364	0.974379	0.974319
2	Naive Bayes	0.923484	0.926255	0.923484	0.922537

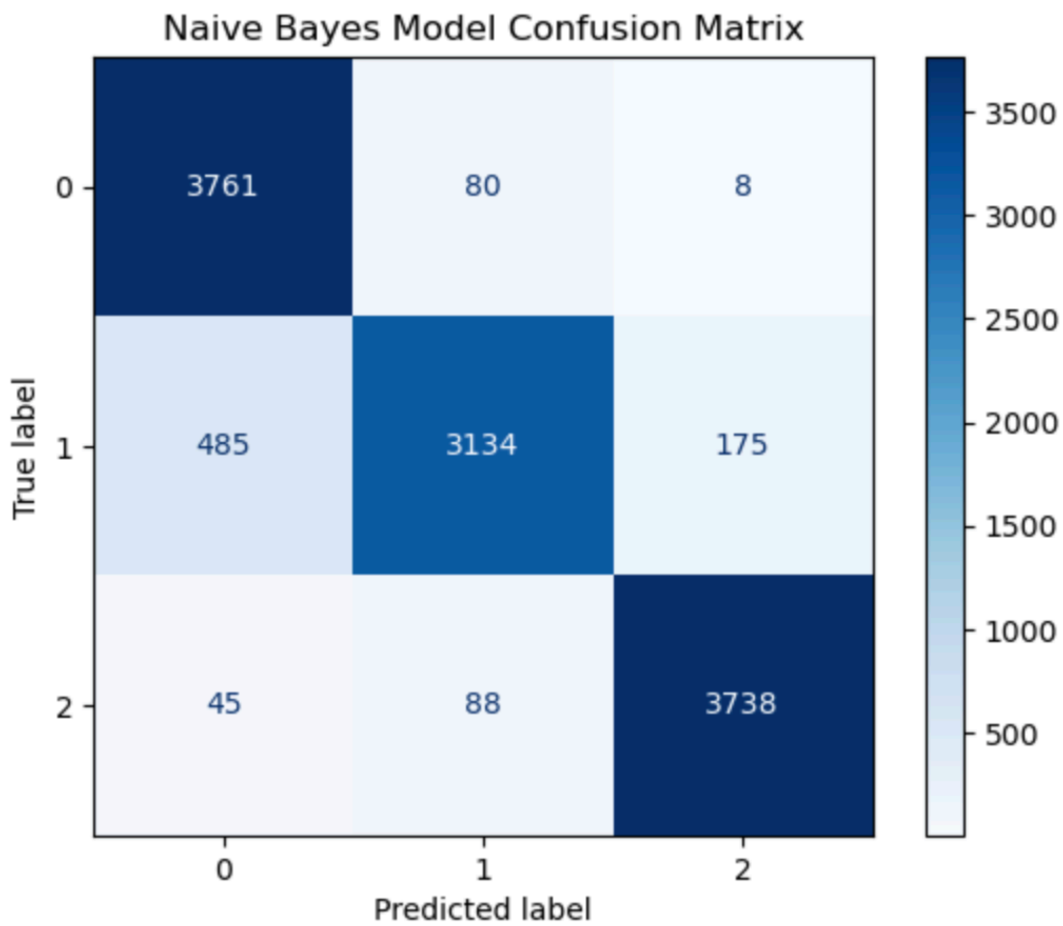
6.1 Üç Modelin Sahip olduğu Metrikler



6.2 Confusion Matrixes

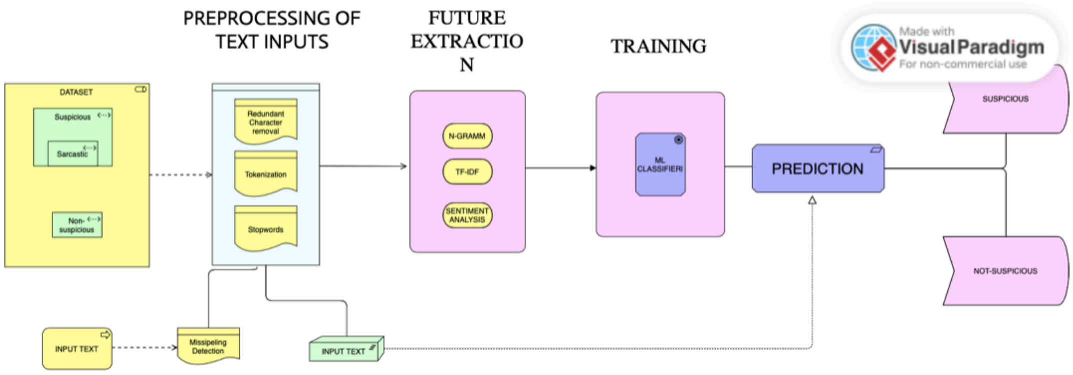






As a result, it got the best ***Acurracy*** rate with ***Tensorflow***'s ***0.984541*** ratio. The approach applied in this way has been more successful than the previous approaches we have encountered.

MODEL DIAGRAM



REFERENCES::

[1]Detecting Suspicious Texts Using Machine Learning Techniques

Omar Sharif 1 , Mohammed Moshiul Hoque 1,* , A. S. M. Kayes 2,* , Raza Nowrozy 2,3 and Iqbal H. Sarker 1

[2] Iskandar, B. Terrorism detection based on sentiment analysis using machine learning. J. Eng. Appl. Sci. 2017, 12, 691–698.

[3]Sarker, I.H. A machine learning based robust prediction model for real-life mobile phone data. Internet Things 2019, 5, 180–193. [CrossRef]

[4] Syed abbas Raza Saidi https://www.kaggle.com/datasets/syedabbasraza/suspicious-tweets

[5]Jiang, M.; Cui, P.; Faloutsos, C. Suspicious behavior detection: Current trends and future directions. IEEE Intell. Syst. 2016, 31, 31–39. [CrossRef]