Artifical Intelligence and Expert System(IT– 306)

PROJECT REPORT

SUBMITTED BY : -

VARUN KUMAR (2K19 / IT / 140)

YASHIT KUMAR (2K19 / IT / 149)

Under the Supervision of : -

Prof. Reena Tripathi

SMS SPAM DETECTION

INTRODUCTION

- In today's globalized world, sms is a primary source of communication. This communication can vary from personal, business, corporate to government.

- With the rapid increase in email usage, there has also been increase in the SPAM sms. SPAM sms, also known as junk sms involves nearly identical messages sent to numerous recipients by email.

- Apart from being annoying, spam sms can also pose a security threat to computer system. It is estimated that spam cost businesses on the order of $100 billion in 2007.

- In this project, we use text preprocessing using natural language processing to perform spam filtering to use emails effectively.

- We try to identify patterns using Different classification algorithms to enable us classify the sms as HAM or SPAM. Mainly using Naïve Byes classifier which is working best to predict spam or not.

The main objectives of this Project are as follows:

- To apply data preprocessing and preparation techniques in order to obtain clean data
- Creating word dictionary and Feature extraction process
- To build machine learning models able to predict whether message is spam or not.
- To analyze and compare models performance in order to choose the best model

IMPLEMENTATION DETAILS

Datasets : -

The Dataset we used is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

Dataset contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

1. Data Cleaning and Analysis : -

We are going to make use of NLTK for processing the messages, WordCloud and matplotlib for visualization and pandas for loading data, NumPy for generating random probabilities for train-test split.

We do not require the columns 'Unnamed: 2', 'Unnamed: 3' and 'Unnamed: 4', so we remove them. We rename the column 'v1' as 'label' and 'v2' as 'message'. 'ham' is replaced by 0 and 'spam' is replaced by 1 in the 'label' column. Finally we obtain the following dataframe.

2. Text Preprocessing : -

Text preprocessing is a method to clean the text data and make it ready to feed data to the model.

Before starting with training we must preprocess the messages. So we perform the following steps: -

- Lower case
- Tokenization
- Removing special characters
- Removing stop words and punctuation
- Stemming

First of all, we shall make all the character lowercase. This is because 'free' and 'FREE' mean the same and we do not want to treat them as two different words.

Then we tokenize each message in the dataset. Tokenization is the task of splitting up a message into pieces and throwing away the punctuation characters. The words like ‘go’, ‘goes’, ‘going’ indicate the same activity.

We can replace all these words by a single word ‘go’. This is called stemming. We are going to use Porter Stemmer, which is a famous stemmingalgorithm.

We then move on to remove the stop words. Stop words are those words which occur extremely frequently in any text. For example words like ‘the’, ‘a’, ‘an’, ‘is’, ‘to’ etc. These words do not give us any information about the content of the text. Thus it should not matter if we remove these words for the text.

3. Model Building : -

We trained various models namely Naive Bayes classifier , Support Vector Machines (SVM) etc. Naive Bayes classifier is a conventional and very popular method for document classification problem.

It is a supervised probabilistic classifier based on Bayes theorem assuming independence between every pair of features. SVMs are supervised binary classifiers which are very effective when you have higher number of features.

The goal of SVM is to separate some subset of training data from rest called the support vectors (boundary of separating hyper-plane). The decision function of SVM model that predicts the class of the test data is based on support vectors and makes use of a kernel trick.

Once the classifiers are trained, we can check the performance of the models on test-set. We extract word count vector for each mail in test-set and predict its class(ham or spam) with the trained NB classifier and SVM model

4. Deployment : -

We deployed our machine learning model on web using streamlit framework. Now User can check spam just by copy pasting their sms with good UI.

RESULTS

In [192]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

In [193]:
```
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
```

In [194]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'ETC': etc,
    'GBDT':gbdt,
}
```

c

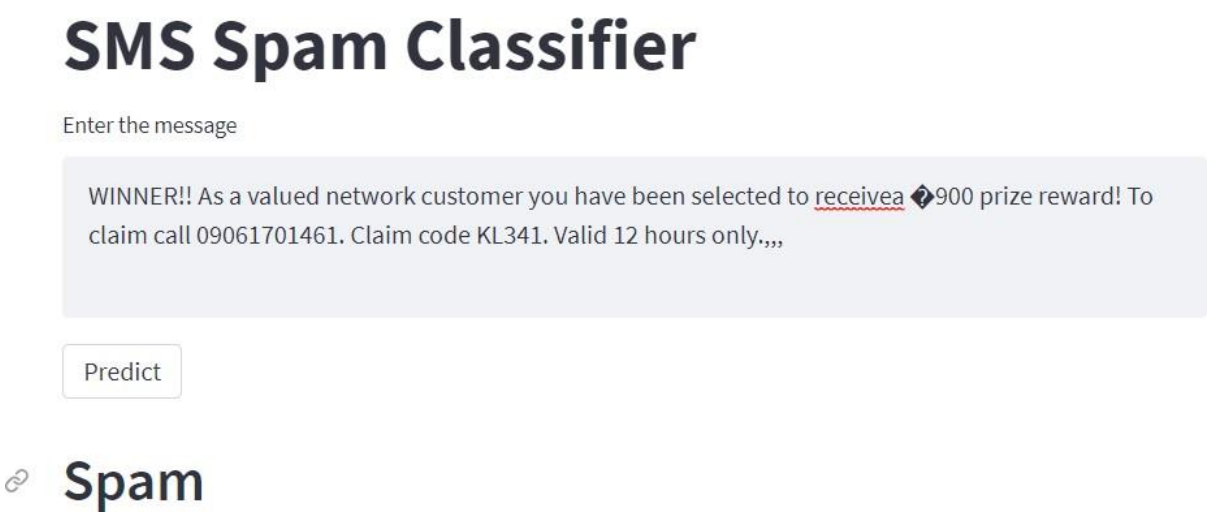
```
In [199]: performance_df
```

Out[199]:

	Algorithm	Accuracy	Precision	Score
1	KN	0.900387	1.000000	0.900387
2	NB	0.959381	1.000000	0.959381
5	RF	0.971954	1.000000	0.971954
7	ETC	0.972921	0.982456	0.972921
0	SVC	0.972921	0.974138	0.972921
6	AdaBoost	0.961315	0.945455	0.961315
4	LR	0.951644	0.940000	0.951644
8	GBDT	0.952611	0.923810	0.952611
3	DT	0.935203	0.838095	0.935203

For text vectorization, First tried bag of words i.e count vectorizer and then we tried Term Frequency Inverse Document Frequency (tfidf) vectorizer and it is giving better results

Various Classification algorithms are used to achieve better results. Out of all algorithms Multinomial Naïve Bayes algorithm is performing best and giving result with accuracy of 0.95 and precision of 1.00.



REFERENCES

Source Code - https://github.com/varunkmr038/Sms-Spam-Detection

Dataset: https://www.kaggle.com/uciml/sms-spam-collection-dataset

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/

https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a

https://www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/

https://www.youtube.com/watch?v=PrkiRVcrxOs