How Artificial Intelligence can determine if a text message is ham or spam

Tracyrenee Follow

Mar 23 · 5 min read ★

As computer users, we are inundated with communications on our phone, particularly in the form of text messages. Some text messages are factual, or ham, but others are not, referred to as spam. It is important for a computer to determine which text messages are spam and which are not, and this is where artificial intelligence comes in. A computer program can be written to look at the text messages and to determine which ones are spam so they can be filtered out by another computer program.

Kaggle, a data science and machine learning website, owned by Google, is one such platform that helps people to develop models to determine if a text message is genuine or spam, and this information can be used to develop programs to filter those spam messages out so the user does not have to see them. In order to accomplish this, Kaggle has in its computer memory many datasets, with one such dataset being the SMS Spam Collection dataset, with the link being here:- SMS Spam Collection Dataset | Kaggle

The SMS Spam Collection 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.

The text messages in this dataset use the concept of natural language processing, or NLP. NLP refers to the branch of artificial intelligence concerned with giving computers the ability to understand text in much the same way that humans can.

The tool in Python that deals with NLP is the Natural Language Toolkit, or NLK. NLTK is an open source collection of libraries, programs and education resources for building NLP programs.

One use of NLP technology is spam detection. The best spam detection technologies use NLP's text classification capabilities to scan text messages for language that often includes spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names and more.

I have written this program in Kaggle's free online Jupyter Notebook, which is similar to Google Colab. The only real difference being the os library is used to bring up the .csv files that are stored in Kaggle's directory.

The first thing I did was to import the libraries that I would need, being numpy for algebraic operations, pandas to manipulate the dataframe, and matplotlib and seaborn for graphical operations. As stated above, I also imported os to bring out the .csv file that contained the text messages and stated whether they are ham or spam:-

Import libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/sms-spam-collection-dataset/spam.csv
```

I then loaded the file, but since it was encoded, I had to add and encoding parameter to get pandas to actually read the file:-

Load file

```
# load datasets
df = pd.read_csv('/kaggle/input/sms-spam-collection-dataset/spam.csv', encoding
="ISO-8859-1")
df
```

When the file was read and converted to a dataframe, I saw there were three unnamed columns, which I decided to drop:-

Drop columns

```
df.drop(['Unnamed: 2', 'Unnamed: 3','Unnamed: 4'], axis=1, inplace=True)
df
```

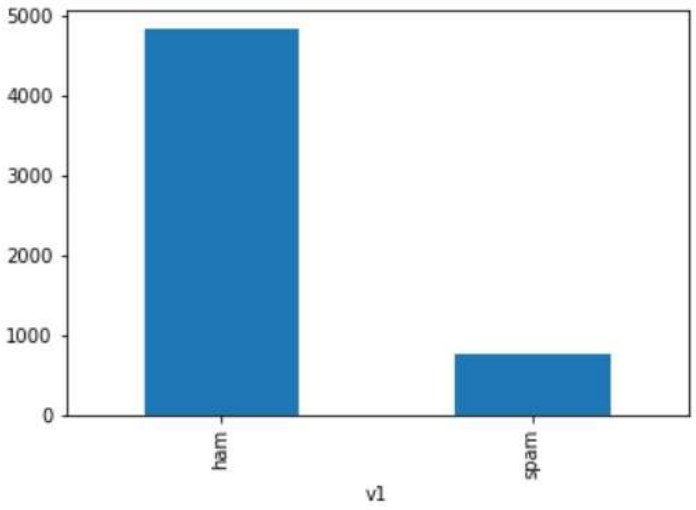
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...

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

5572 rows × 2 columns

I analysed the target variable, vi, and found there is a slight class imbalance in this dataset:-

```
]:
df.groupby('v1').v1.count().plot.bar(ylim=0)
plt.show()
```



I decided to map the target variable, vi, by creating a dictionary and assigning 1 to ham and 0 to spam:-

Map v1

```
]:
dic = {'ham':1 ,'spam':0}
df.v1 = df.v1.map(dic)
```

```
df
```

]:

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

I then created a new column where all of the processed text would be placed:-

Preprocess raw text and get ready for machine learning

:

```
#create new column
df['processedtext'] = df['v2']
df
```

]:

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

5572 rows × 3 columns

I imported the nltk library, which would enable text preprocessing to take place. The nltk library stemmed the words and removed special characters:-

```
]:
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
nltk.download('stopwords')
import re
import warnings
warnings.filterwarnings('ignore')

stemmer = PorterStemmer()
words = stopwords.words("english")

df['processedtext'] = df['processedtext'].apply(lambda x: " ".join([stemmer.stem
(i)
for i in re.sub("[^a-zA-Z]", " ", x).split() if i not in words]).lower())
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

After the words had been stemmed and the special characters had been removed, I converted all of the processed text to lower case, removed any special characters, and removed any words less than three characters in length:-

```
]:
#make all words lower case
df['processedtext'] = df['processedtext'].str.lower()

# remove special characters, numbers, punctuations
df['processedtext'] = df['processedtext'].str.replace("[^a-zA-Z#]", " ")

#remove words less than 3 characters
df['processedtext'] = df['processedtext'].apply(lambda x: ' '.join([w for w in
x.split() if len(w)>3]))
```

When the processed text had been fully preprocessed, I defined the X and y variables. The target variable, df,v1 became y. The independent variable, df.processedtext became X:-

Define X and y variables

```
#define X and y
y = df['v1']
X = df['processedtext']
```

Once X and y had been defined, I used sklearn's TfidVectorizer() function to vectorize the X variable:-

Convert text to word frequency vectors

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf = TfidfVectorizer(stop_words='english', max_df=0.7)
df_tfIdf = vectorizer_tfidf.fit_transform(X.values.astype('U'))
print(vectorizer_tfidf.get_feature_names()[:10])
```

```
['aaniy', 'aaoooooright', 'aathi', 'abbey', 'abdomen', 'abeg', 'abel', 'aberde
en', 'abil', 'abiola']
```

I then split the X and y variables up into training and validating sets using sklearn's train_test_split() function:-

Split X for training and validation

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_val, y_train, y_val = train_test_split(df_tfIdf, y, test_size=0.10, r
andom_state=1, shuffle=True)
X_train.shape, X_val.shape, y_train.shape,y_val.shape
```

```
((5014, 5126), (558, 5126), (5014,), (558,))
```

Once the X and y variables had been split into training and validation datasets, I selected the model. The PassiveAgressiveClassifier() function is well suited to NLP, so I chose that. When I trained and fitted the model, I achieved 99.86% accuracy:-

Select model

```
from sklearn.linear_model import PassiveAggressiveClassifier

model = PassiveAggressiveClassifier(max_iter=1000, random_state=1,tol=1e-3).fit
(X_train, y_train)
print(model.score(X_train, y_train))
```

```
0.998603909054647
```

I then predicted on the validation set and achieved 97.67% accuracy. I created a confusion matrix and 11 examples were incorrect for ham, while 2 examples were incorrect for spam :-

Predict on validation set

```
y_pred = model.predict(X_val)
print(model.score(X_val, y_val))
```

```
0.9767025089605734
```

Confusion Matrix

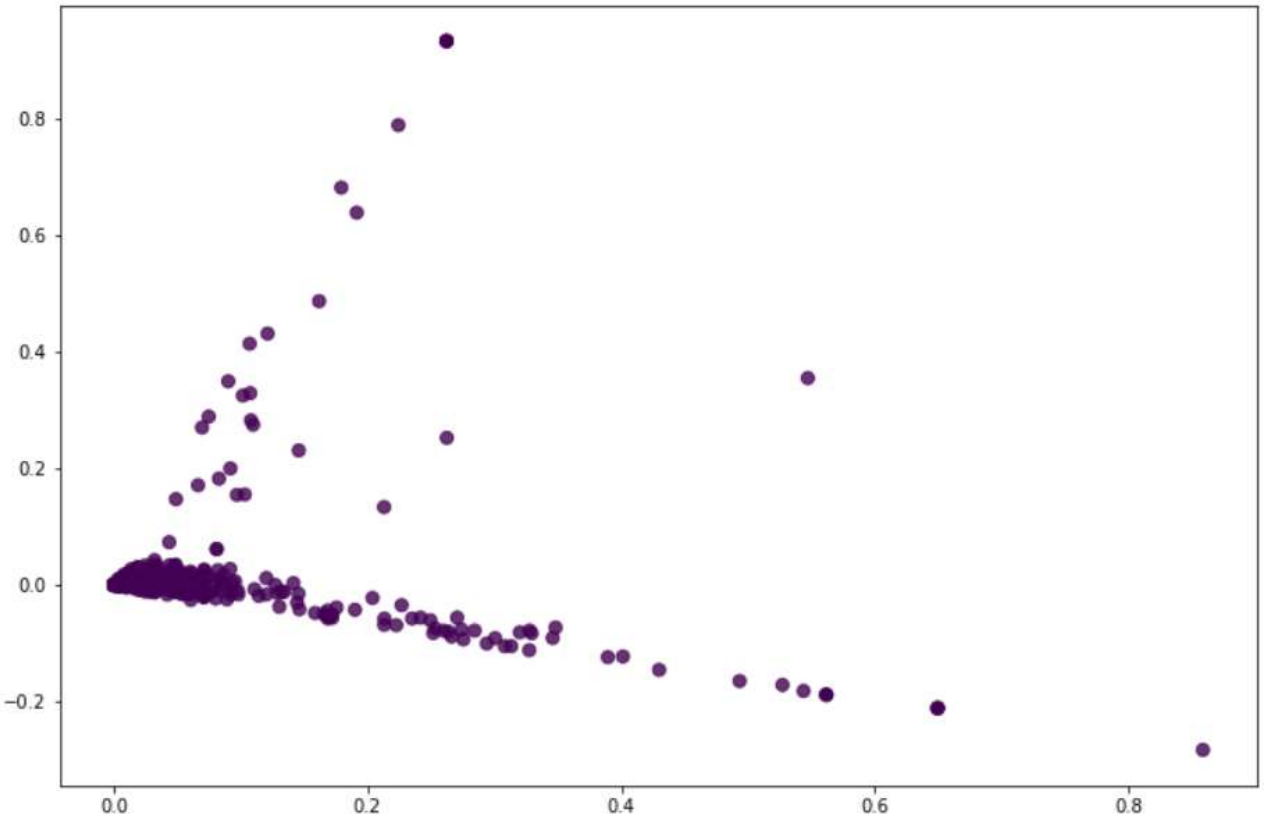
Confusion Matrix

```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_val,y_pred))
```

```
[[ 57  11]
 [  2 488]]
```

Finally, I plotted the predicted examples on a graph, using the TruncatedSVD() function, which is suited to working with sparse data:-



The code for this program can be found in its entirety in my personal Kaggle account, the link being here:- Spam Detection | Kaggle

Python Artificial Intelligence Nltk Data Science Machine Learning