

# Spam Detection

March 22, 2023

## 1 SMS Spam Detection

1.0.1 by Jason Paul Miller

03/14/23 PI Day

## 2 Important Imports

```
[ ]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
pd.options.mode.chained_assignment = None # removes false positive warning
import matplotlib.pyplot as plt
import re # Text cleanup
import string # Text cleanup
from gensim.models import Word2Vec
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Conv1D, GlobalMaxPooling1D, Dense
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
from sklearn.feature_extraction.text import CountVectorizer
```

### 2.0.1 Read in the data

```
[ ]: data = pd.read_csv('spam.csv', encoding='latin-1')
data
```

```
[ ]:      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]

## 2.0.2 Clean up the data

```

[ ]: # Remove unnecessary columns
data = data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis = 1)
print(data.shape)
data

```

(5572, 2)

```

[ ]:      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 İ_ 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 x 2 columns]

```

[ ]: # Remove any duplicates
data = data.drop_duplicates()
print(data.shape)

```

(5169, 2)

```
[ ]: def clean_text(text):
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))

    # Remove digits
    text = re.sub(r'\d+', '', text)

    # Convert to lowercase
    text = text.lower()

    return text

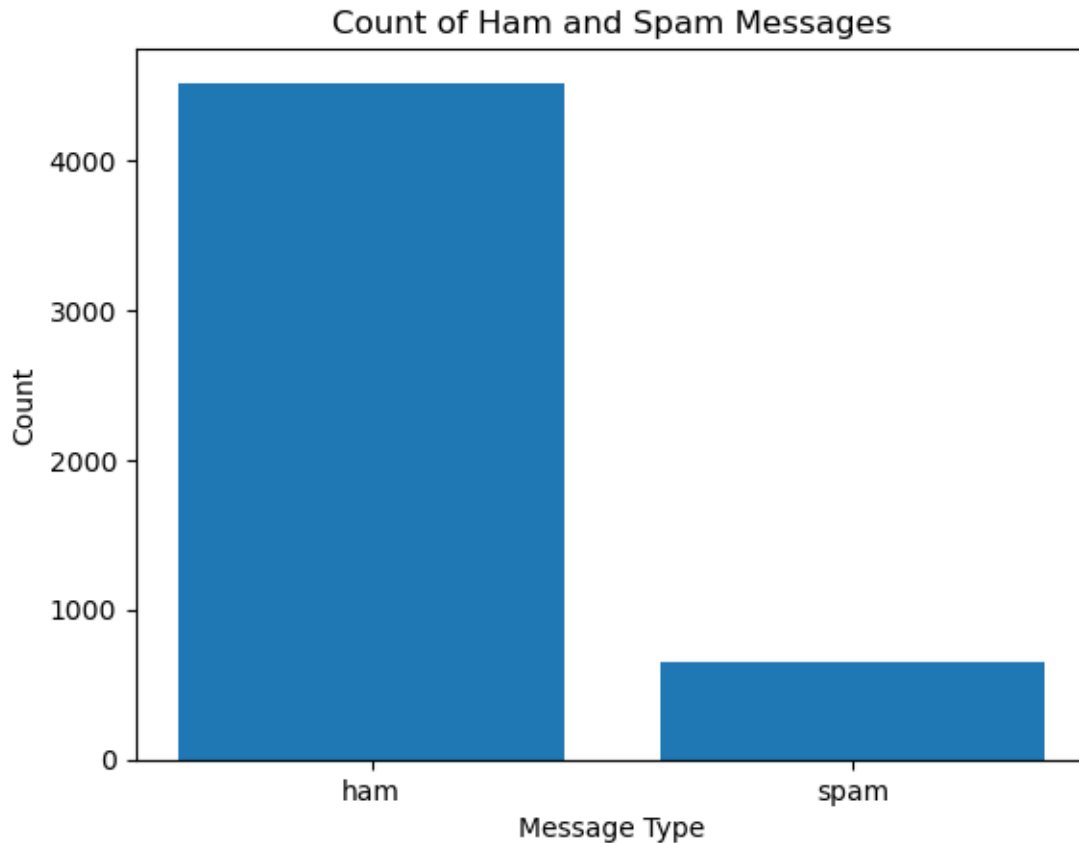
# Text Cleaning
data.loc[:, 'v2'] = data['v2'].apply(clean_text)
```

### 2.0.3 Visualize the data

```
[ ]: # Count the number of 'ham' and 'spam' messages
ham_count = (data['v1'] == 'ham').sum()
spam_count = (data['v1'] == 'spam').sum()

# Plot a bar chart of the counts
labels = ['ham', 'spam']
counts = [ham_count, spam_count]
plt.bar(labels, counts)
plt.xlabel('Message Type')
plt.ylabel('Count')
plt.title('Count of Ham and Spam Messages')
```

```
[ ]: Text(0.5, 1.0, 'Count of Ham and Spam Messages')
```



#### 2.0.4 Split the Data

```
[ ]: # Randomize the rows
data = data.sample(frac=1, random_state=42)

# Split the data into training and test sets
train_data = data[:int(0.9*len(data))]
test_data = data[int(0.9*len(data)):]

# Print the shapes of the training and test sets
print('Training data shape:', train_data.shape)
print('Test data shape:', test_data.shape)
```

Training data shape: (4652, 2)

Test data shape: (517, 2)

## 2.0.5 Word Embedding

```
[ ]: # Train the model
model = Word2Vec(train_data['v2'], vector_size=100, window=5, min_count=1,
    ↪workers=4)

# Create a dictionary that maps each word to its corresponding vector
    ↪representation
word_vectors = model.wv

# Convert each text into a fixed-length vector by taking the mean of the word
    ↪vectors
text_vectors = []
for text in train_data['v2']:
    words = text.split()
    vectors = []
    for word in words:
        if word in word_vectors.key_to_index:
            vectors.append(word_vectors.get_vector(word))
    if len(vectors) > 0:
        text_vector = np.mean(vectors, axis=0)
    else:
        text_vector = np.zeros(100)
    text_vectors.append(text_vector)

X_train = np.array(text_vectors)
y_train = np.array([1 if label == 'spam' else 0 for label in train_data['v1']])
```

Do the same for the test data

```
[ ]: # Train the model
model = Word2Vec(test_data['v2'], vector_size=100, window=5, min_count=1,
    ↪workers=4)

# Create a dictionary that maps each word to its corresponding vector
    ↪representation
word_vectors = model.wv

# Convert each text into a fixed-length vector by taking the mean of the word
    ↪vectors
text_vectors = []
for text in test_data['v2']:
    words = text.split()
    vectors = []
    for word in words:
        if word in word_vectors.key_to_index:
            vectors.append(word_vectors.get_vector(word))
```

```

    if len(vectors) > 0:
        text_vector = np.mean(vectors, axis=0)
    else:
        text_vector = np.zeros(100)
    text_vectors.append(text_vector)

X_test = np.array(text_vectors)
y_test = np.array([1 if label == 'spam' else 0 for label in test_data['v1']])

```

## 2.0.6 Train a CNN

```

[ ]: # Create a simple CNN
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
    ↪input_shape=(X_train.shape[1], 1)))
model.add(GlobalMaxPooling1D())
model.add(Dense(units=1, activation='sigmoid'))

# Reshape X to a 3D numpy array with shape (num_samples, num_timesteps,
    ↪num_features)
X_resaped = np.array(X_train).reshape((X_train.shape[0], X_train.shape[1], 1))

# Train the model
model.compile(loss='binary_crossentropy', optimizer='adam',
    ↪metrics=['accuracy'])
model.fit(X_resaped, y_train, batch_size=32, epochs=10, validation_split=0.2)

```

```

Epoch 1/10
117/117 [=====] - 1s 4ms/step - loss: 0.5218 -
accuracy: 0.8546 - val_loss: 0.4239 - val_accuracy: 0.8668
Epoch 2/10
117/117 [=====] - 0s 3ms/step - loss: 0.4039 -
accuracy: 0.8705 - val_loss: 0.4013 - val_accuracy: 0.8668
Epoch 3/10
117/117 [=====] - 0s 3ms/step - loss: 0.3909 -
accuracy: 0.8705 - val_loss: 0.3930 - val_accuracy: 0.8668
Epoch 4/10
117/117 [=====] - 0s 3ms/step - loss: 0.3833 -
accuracy: 0.8705 - val_loss: 0.3859 - val_accuracy: 0.8668
Epoch 5/10
117/117 [=====] - 0s 2ms/step - loss: 0.3774 -
accuracy: 0.8705 - val_loss: 0.3812 - val_accuracy: 0.8668
Epoch 6/10
117/117 [=====] - 0s 2ms/step - loss: 0.3733 -
accuracy: 0.8705 - val_loss: 0.3779 - val_accuracy: 0.8668
Epoch 7/10
117/117 [=====] - 0s 2ms/step - loss: 0.3711 -

```

```

accuracy: 0.8705 - val_loss: 0.3757 - val_accuracy: 0.8668
Epoch 8/10
117/117 [=====] - 0s 2ms/step - loss: 0.3682 -
accuracy: 0.8705 - val_loss: 0.3752 - val_accuracy: 0.8668
Epoch 9/10
117/117 [=====] - 0s 2ms/step - loss: 0.3666 -
accuracy: 0.8705 - val_loss: 0.3712 - val_accuracy: 0.8668
Epoch 10/10
117/117 [=====] - 0s 2ms/step - loss: 0.3640 -
accuracy: 0.8705 - val_loss: 0.3694 - val_accuracy: 0.8668

```

```
[ ]: <keras.callbacks.History at 0x286876bf0>
```

## 2.0.7 Test our model

```
[ ]: # Make predictions on the test set
y_pred = model.predict(X_test)
y_pred = np.round(y_pred).flatten()

# Calculate the MSE
print(classification_report(y_test, y_pred))
```

```

17/17 [=====] - 0s 506us/step
              precision    recall  f1-score   support

         0           0.91         1.00         0.95         470
         1           0.00         0.00         0.00          47

 accuracy                   0.91         517
 macro avg           0.45         0.50         0.48         517
 weighted avg           0.83         0.91         0.87         517

```

```

/Users/jasonmiller/miniforge3/envs/myenv/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/jasonmiller/miniforge3/envs/myenv/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/Users/jasonmiller/miniforge3/envs/myenv/lib/python3.10/site-
packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

```

### 3 Trying another approach

```
[ ]: # Splitting the data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data['v2'], data['v1'],
    ↪ test_size = 0.1, random_state = 0)

# Convert the text data into a matrix of token counts
cv = CountVectorizer()
X_train_count = cv.fit_transform(X_train.values)
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]])
```

```
[ ]: # Fitting multinomial naive bayes
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count, y_train)

# finding accuracy of the test dataset
X_test_count = cv.transform(X_test)
model.score(X_test_count, y_test)
```

```
[ ]: 0.9922630560928434
```