SPAM VS. HAM

Machine Learning

By Juliane Tess

Overview

- Brief personal biography
- Significance of the study
- Analysis
- Limitations
- Future Recommendations

From Teacher to Analyst…

Bachelors in Theatre Arts

Teacher & Studio Director

Studio Owner?

Next step =

creative outlet + love of math

Entity: Data Science Program

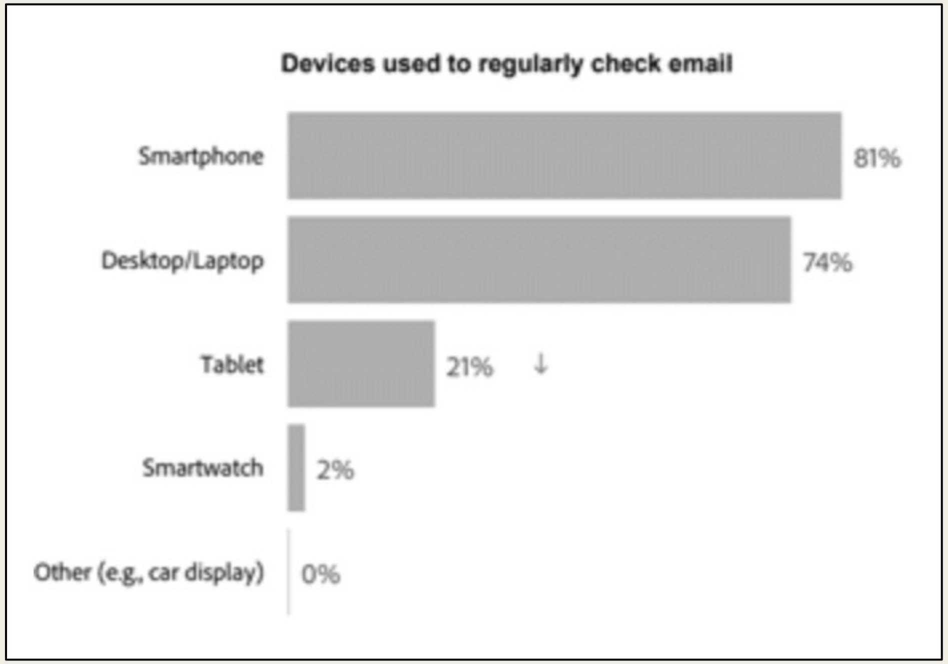
Over the course of 33 weeks…

- Hard, tech skills:
  - *Coding languages, such as, Python, R, SQL*
  - *Data Wrangling and Visualization (Tableau, Excel, etc.)*
  - *Project Management*
  - *Big Data*
  - *Machine Learning and Modeling*
- Soft skills

Significance

- High demand and large user base
- As of 2020, among most popular digital activities in US
- Mobile most popular
- E-mail newsletters used by B2B & B2C marketers

(Source: Statista)



(Source: EmailMonday)

Significance (cont.)

1

Reduce exploitation of users and their data

- Anti-malware tool

2

Reduce Fraud

3

Reduce lost revenue

SPAM VS. HAM

CASE STUDY

Explore the data

In [4]:
```
sms.head()
```

Out[4]:

	label	message
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...

In [5]:
```
print(len(sms))
```

```
5572
```

In [7]:
```
sms.groupby('label').describe()
```

Out[7]:

	message			
	count	**unique**	**top**	**freq**
label				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

Utilized dataframe with the length of 5572

- *4825 = ham*
- *747 = spam*

PREPROCESS:

1. Remove Punctuations

```
In [12]: sample_sms = 'Sample message!...'
         no_punc = [char for char in sample_sms if char not in string.punctuation]
         no_punc = ''.join(no_punc)
         print(no_punc)

         Sample message
```

2. Remove Stopwords

```
In [13]: stopwords.words('english')[0:5]
Out[13]: ['i', 'me', 'my', 'myself', 'we']
```

3. Split words into individual

```
In [14]: no_punc.split()
Out[14]: ['Sample', 'message']
```

4. Lowercase Words

```
In [15]: lower_sms = [word.lower() for word in no_punc.split() if word.lower() not in stopwords.words('english')]

In [16]: lower_sms
Out[16]: ['sample', 'message']
```

5. Combine all into a function to apply to dataframe later:

```
In [17]: def text_clean(sample_sms):
             no_punc = [char for char in sample_sms if char not in string.punctuation]
             no_punc = ''.join(no_punc)
             return [word.lower() for word in no_punc.split() if word.lower() not in stopwords.words('english')]

In [18]: sms2 = sms['message'].head().apply(text_clean)

In [19]: sms2.head()
Out[19]: 0    [go, jurong, point, crazy, available, bugis, n...
         1                        [ok, lar, joking, wif, u, oni]
         2    [free, entry, 2, wkly, comp, win, fa, cup, fin...
         3        [u, dun, say, early, hor, u, c, already, say]
         4    [nah, dont, think, goes, usf, lives, around, t...
         Name: message, dtype: object
```

```
In [34]: x = sms['message']
         y = sms['label']

In [35]: x_train, x_test, y_train, y_test = train_test_split(
             x,y,test_size = .2, random_state=101)
```

```
In [37]: pipeline = Pipeline([
             ('bow', CountVectorizer(analyzer=text_clean)),
             ('tfidf', TfidfTransformer()),
             ('classifier', MultinomialNB()),
         ])

In [38]: pipeline.fit(x_train,y_train)

Out[38]: ▸ Pipeline
           ▸ CountVectorizer
           ▸ TfidfTransformer
           ▸ MultinomialNB

In [39]: predictions = pipeline.predict(x_test)

In [40]: print(classification_report(predictions,y_test))

                  precision    recall  f1-score   support

          ham          1.00      0.95      0.98      1027
         spam          0.65      1.00      0.79        88

     accuracy                              0.96      1115
    macro avg          0.83      0.98      0.88      1115
 weighted avg          0.97      0.96      0.96      1115
```

```
In [41]: print("Score:", pipeline.score(x_test, y_test))

         Score:  0.957847533632287
```

- This means the model is accurate approximately 95% of the time.

MACHINE LEARNING: TRAIN TEST SPLIT

Goal: to accurately predict which of the messages are *spam* and which are *ham*

Project Limitations

- Stem/Lemmatization
  - *NLTK has lots of built-in tools and great documentation on a lot of methods of normalization. These tools—such as Porter Stemmer-–are not always great for using with abbreviations or shorthand (a.k.a. slang), as was prevalent within the data frame chosen for the case study.*

Machine Learning Limitations: *Email Spam Filtering*

- Difficult to mine effectively-represented features
- Lack of security strategy against attack
  - *Ex: causative or exploratory, targeted or indiscriminate*
- Algorithm shortcomings:
  - *Need for knowledge from expert in a particular field*
  - *Dimensionality*
  - *High computational cost*
- Other open research problems

(Source: ScienceDirect)

Future Recommendations:
Deep Learning

Future of email spam filters

Deep Learning Advantage

Number of available training data increasing

Use intricate and huge models

Deep Learning Imperfections

Thank you for your consideration!

QUESTIONS?

Please feel free to click on the hyperlink for my LinkedIn or find me on Slack.

LINKEDIN

GITHUB

SLACK