Machine Learning Nanodegree

Capstone Proposal

Mohamed Ahmed Elhwary

November 27, 2018

Definition

1- Project Overview

Email is one of the most important forms of communication. "Projections show that by the end of 2019, we can expect to see 2.9 billion worldwide email users (which is more than one-third of the global population). In regard to email sends per day worldwide, we know that about 269 billion emails were sent and received each day in 2017. That figure is expected to grow to almost 320 billion daily emails in 2021, according to Statista". I want to help people with this project, as it can classify incoming e-mails to spam or not spam emails so they can save their time and privacy.

Daily Email Traffic	2015	2016	2017	2018	2019
Total Worldwide Emails Sent/Received Per Day (B)	205.6	215.3	225.3	235.6	246.5
% Growth		*5%*	*5%*	*5%*	*5%*
Business Emails Sent/Received Per Day (B)	112.5	116.4	120.4	124.5	128.8
% Growth		*3%*	*3%*	*3%*	*3%*
Consumer Emails Sent/Received Per Day (B)	93.1	98.9	104.9	111.1	117.7
% Growth		*6%*	*6%*	*6%*	*6%*

Table 2: Worldwide Daily Email Traffic (B), 2015-2019

2- Problem Statement

In this project I will use supervised learning to learn a classifier to classify the coming messages (e-mails) to spam or not spam. As now days the e-mails and SMS messages ar so common and people receive them every day. I want to prevent the spam messages to be received to users. I have a dataset with 5572 labeled messages.

1- Loading the data and exploring it.
2- Regular Expressions to replace url, e-mail address, and phone numbers.
3- Pre-processing:
- Sentence Segmentation.
- Word Tokenization.
- Text Lemmatization.
- Identifying Stop Words.

4- Feature engineering.
5- Building the models.
6- Evaluation.

Metrics

Accuracy: is a common metric for binary classifiers; it takes into account both true positives and true
negatives with equal weight.

$$\text{Accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{dataset size}}$$

This metric was used when evaluating the classifier because false negatives and false positives both erode the user experience.

F-Score: which is defined as the harmonic mean of precision and recall. I found F1_score as
appropriate to use as report metric in order to have a good idea of how the

algorithm is :

$$F1 = \frac{2 P * R}{P + R}$$

Analysis

3- Analysis

Below describes how the data was gathered, which features were selected, and which algorithms
were explored. Finally, I outline the benchmark used to evaluate the performance of the trading
strategy .

- Data exploration

The dataset used for this project is SMS Spam Collection dataset originates from the UCI Machine Learning Repository. This dataset has been collected from free or free for research sources at the Internet. The collection is composed of just one text file, where each line has the correct class followed by the raw message. Dataset does not require any kind of cleaning, wrangling and there is no null value in any column .

In [2]:
```
df = pd.read_csv('data-spam-ham.csv',  usecols=[0, 1])
df.head()
```

Out[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...

In [3]:
```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
v1    5572 non-null object
v2    5572 non-null object
dtypes: object(2)
memory usage: 87.1+ KB
```

Out[3]:

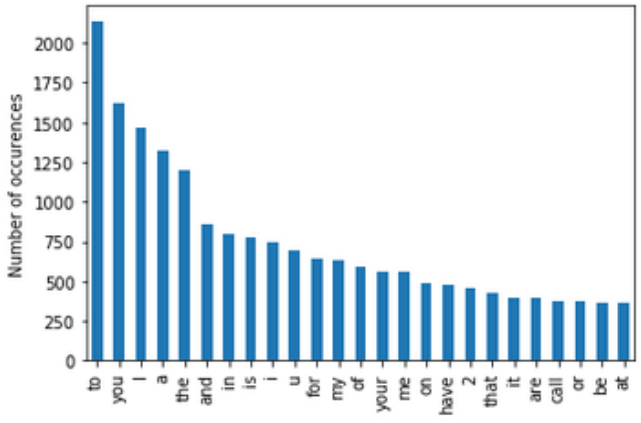
	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

- Data Visualization

There are a lot of words are common in the same class of messages, Since each category of messages has some similar type of keywords i.e. in spam message words like Free, Winner, win, won, award, congrats, congratulation, selected, urgent, and Cash it is easier to classify them and in ham message some words like go , ok ,... so I will display graph demonstrate most

frequency words that appear in data set to make it easy to understand which words common in both two types and avoid them in prediction .so I will use Techniques which we are going to us are based on word count and term-frequency inverse document-frequency (tf-idf) transform .

In [4]:
```
# Visualization of the most frequent words of the dataset
from collections import Counter
import matplotlib.pyplot as plt

most_common_cnt = Counter(" ".join(df["v2"]).split()).most_common(25)
df_most_common = pd.DataFrame.from_dict(most_common_cnt)
df_most_common = df_most_common.rename(columns={0: "words", 1 : "count"})
fig = plt.figure()
ax = fig.add_subplot(111)
df_most_common.plot.bar(ax=ax, legend = False)
xticks = np.arange(len(df_most_common["words"]))
ax.set_xticks(xticks)
ax.set_xticklabels(df_most_common["words"])
ax.set_ylabel('Number of occurences')
plt.show()
```



- a lot of stop words need to be removed
- the most effective words not shown well so we need other common words.

Methodology

Algorithms and Techniques

Naive Bayse is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. The advantage

of using this classifier is that it work good on both numeric as well as textual data and moreover it is easier to implement. The disadvantage of this classifier is that its performance gets poorer when the extracted features are correlated to each other.

Decision Tree is another classifier algorithm which is widely used for the purpose of classification. It works on a series of some test questions & conditions applied on it. It is represented in a tree form of structure where the branches of tree represents “weight” and each leaf is a different “class”. Decision- tree is good in learning disjunction expressions and can handle noisy data. But training a decision-tree may act as an expensive process. If there is a mistake in very higher level then there will be flaws in whole sub-tree and whole structure may act as invalid .
https://scikit-learn.org/stable/modules/tree.html

Logistic Regression is another way to determine a class label, depending on the features. Logistic regression takes features that can be continuous (for example, the count of words in an email) and translate them to discrete values (spam or not spam).
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

SVM is an algorithm of supervised learning which is used for both classification and regression. In most of cases it is used as a classifier. It is used for many NLP tasks in text classification. Here each data item is plot as n-dimensional space where n represents no. of features extracted. SVM comes with a unique feature that it includes both types: positive & negative training sets. It represents each text-document as a vector where its dimensions are the no. of different keywords. But if the size of text document is large then there will be a number of dimensions in hyper-space which may increase computational cost of the process .

In [20]:

```
# ploating data for F1 Score
y_pos = np.arange(len(objects))
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.7)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Accuracy of Models')
plt.show()


```

4- BenchMark model

The result shows that Naive Bayes work better (not the best) on the dataset with an accuracy_score of 88% and we can use it as a BenchMark model because it does not need tuning. But the highest accuracy is SVM so it is the fit one.

5- Methodology

As specified before dataset does not require any sort of cleaning. Column “v1” contain categorical value ham and spam since classifier require numeric value ham is replaced with 1 and spam is replaced with 0 ,the only things I do here is to ensure that message body does not contain any type of Regular Expression the type of it found in Wikipedia (https://en.wikipedia.org/wiki/Regular_expression) Other thing I do is Data processing To train our classifier we need to use term frequency– inverse document frequency (tf–idf), it is a numerical statistic that is intended to reflect how important a word is to a document in a corpus. It is used as a weighting factor in information retrieval, text mining, and user modelling. tf.idf(t,d)=tf(t,d)×idf(t) The tf-idf value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Remove stop words , all details about it found here (https://pythonspot.com/nltk-stop-words/) This is sample from our data after doing above process :-

```
In [13]:
1 cleaned_messages = clean_data(messages)
2 print(cleaned_messages.head(10))
```

```
0    go until jurong point crazy available only in ...
1                            ok lar joking wif u oni
2    free entry in numbr a wkly comp to win fa cup ...
3          u dun say so early hor u c already then say
4    nah i don t think he goes to usf he lives arou...
5    freemsg hey there darling it s been numbr week...
6    even my brother is not like to speak with me t...
7    as per your request melle melle oru minnaminun...
8    winner as a valued network customer you have b...
9    had your mobile numbr months or more u r entit...
Name: v2, dtype: object
```

- process of stop words removing

```
In [29]:
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

stopset = set(stopwords.words("english"))
vectorizer = TfidfVectorizer(stop_words=stopset,binary=True)
```

6- Implementation And Results

All classification algorithms needs all the data to be numeric, this was handled during the preprocessing step and it was mentioned in the above section.
Pre-processing is the actual first and most important step that needs to be handled carefully and with patience as it takes 80% of the work time on the project, preprocessing the data well leads to a better results in the prediction of any model. After analyzing the data to explore trends or it's characteristics, I have preprocessed the data to be ready and suitable for modeling and prediction. The final output of the preprocessing phase is a clean dataset that's labeled numerically and encoded without any unnecessary feature that doesn't contribute to the prediction of the target variable.
The dataset is then divided into 75% for training and 25% testing data.
The classifiers are initialized for Naive Bayes, Decision Tree, Logistic Regression, and SVM, and the training for the model starts, then the testing outputs the final predictions and compares them with the
test labels to measure the accuracy using the accuracy_score .
https://scikit-learn.org/stable/modules/svm.html

In [19]:

```
# loop to call function for each model
clf = [NaiveBayse,DecisionTree,LogisticRegression,SVC]
names =['NaiveBayse','DecisionTree','LogisticRegression','SVC']
pred_val = [0,0,0,0]
for a in range(0,4):
    train_classifier(clf[a], X_train, y_train)
    y_pred = predict_labels(clf[a],X_test)
    pred_val[a] = f1_score(y_test, y_pred,  average='binary')
    print ("{} Accuracy: {}".format(names[a],pred_val[a]))


```

```
NaiveBayse Accuracy: 0.877323420074
DecisionTree Accuracy: 0.909090909091
LogisticRegression Accuracy: 0.833333333333
SVC Accuracy: 0.923076923077
```

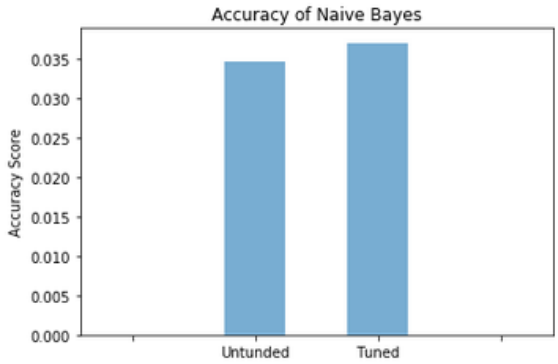
In [20]:

```
# ploating data for F1 Score
y_pos = np.arange(len(objects))
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.7)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Accuracy of Models')
plt.show()


```

Accuracy after and before Tuning

In [21]:

```
# ploating data for Accuracy Score
# ploating data for Accuracy of Models between 1.00 - 0.90 for better visualization
objects = ('','Untunded', 'Tuned','')
y_pos = np.arange(4)
y_val = [0,0.03470790378,0.037062937063,0 ]
plt.bar(y_pos,y_val, align='center',width = 0.5, alpha=0.6)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Accuracy of Naive Bayes')
plt.show()


```



The process Removing Stop words from model make it faster and more accurate than un-tuned one.

```
In [29]: from nltk.corpus import stopwords
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.feature_extraction.text import CountVectorizer

         stopset = set(stopwords.words("english"))
         vectorizer = TfidfVectorizer(stop_words=stopset,binary=True)
```

In Results, we can say that the result is trustworthy and the project is finished under satisfaction the precision of 88%. The only thing negative is the time it takes for training maybe 10 to 15 seconds but the output has correct labels and that is the most important .

7- Justification

The result of the models was very satisfactory, comparing to the benchmark. All the final models worked better than benchmark even with half the dataset, it is possible to say that the result is trustworthy and the project is finished under satisfaction the precision of 88% .

I suggest to use the SVM model as the main classifier for the final program as it provided the highest accuracy.

8- Conclusion

In this project, we tried to analysis different methods to identify spam messages. We used the different approach, based on word count and term-frequency inverse document-frequency transform to classify the messages. Since test data have very high meaning for human and very difficult for the machine to understand, the biggest challenge was to analysis the test data and convert it into some meaningful numeric data without disturbing the relation between categories. Best method to convert the test data into meaningful

numeric data is tf-idf vectorizer. This was the most interesting part of the whole project to convert huge amount of text data into numeric data. Best result was generated using tf-idf vectorizer with SVM and DT classifier, both achieved accuracy approx. 91%, which is a satisfactory result . The process used for this project can be summarized using the following steps:

1- Loading the data and exploring it.
2- Regular Expressions to replace url, e-mail address, and phone numbers.
3- Pre-processing:
    - Sentence Segmentation.
    - Word Tokenization.
    - Text Lemmatization.
    - Identifying Stop Words.
4- Feature engineering.
5- Building the models.
6- Evaluation.

Improvement

Most Models work good so I think if I use CNN will achieve results Better than one that I have now , because Artificial neural-networks work on the concept of human brain consisting neurons. It consists of a layered arrangement of neurons where the input vectors are converted into the some form of output. CNN is considered to be a good classifier because it can better handle multiple categories and work well on it. It supports fast testing phase. CNN is when combined with naive bayes algorithm it comes up with a new idea which is called knowledge based neural networks and this is much efficient in handling noisy data .

Resources

1. Fig1 https://www.campaignmonitor.com/blog/email-marketing/2018/03/shocking-truth-about-how-many-emails-sent/