Spam Detection with Machine Learning

Rapport de projet de fin d'année

Réalisé par : ELASRI Hala

Encadré par Mr BELMEKKI Abdelhamid

Année scolaire 2020/2021 05/07/2021

Introduction

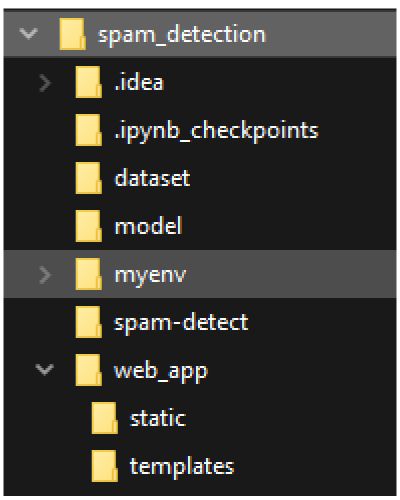
We are going to build an SMS spam detection web application. This application will be built with Python using the Flask framework and will include a machine learning model that we will train to detect SMS spam.

1. Prerequisites

We will need:

- Python 3. The Anaconda distribution includes a number of useful libraries for data science.
- Flask, HTML, and CSS.

2. File Structure



3. Set Up a Python Virtual Environment

Creating a new Python virtual environment

```
C:\Users\viet\spam_detection>conda create -n myenv python=3.6
```

Activating the environment

```
C:\Users\viet\spam_detection>conda activate myenv
```

Creating a new virtual environment

```
(myenv) C:\Users\viet\spam_detection>python -m venv myenv
```

Activating the new environment

```
(myenv) C:\Users\viet\spam_detection>myenv\Scripts\activate
```

Our prompt has been modified to look like the following:

```
(myenv) (myenv) C:\Users\viet\spam_detection>
```

4.Install Required Packages

Next, we will install all the packages needed for this tutorial:

```
(myenv) (myenv) C:\Users\viet\spam_detection>pip  install jupyterlab Flask lightgbm nexmo matplotlib plotly plotly-expre
ss python-dotenv nltk numpy pandas  regex  scikit-learn wordcloud
```

Here are some details about these packages:

- jupyterlab is for model building and data exploration.
- flask is for creating the application server and pages.
- lightgbm is the machine learning algorithm for building our model
- nexmo is a Python library for interacting with your Vonage account
- matplotlib, plotly, plotly-express are for data visualization
- python-dotenv is a package for managing environment variables such as API keys and other configuration values.
- nltk is for natural language operations
- numpy is for arrays computation
- pandas is for manipulating and wrangling structured data.
- regex is for regular expression operations
- scikit-learn is a machine learning toolkit
- wordcloud is used to create word cloud images from text

After installation, we will start our Jupyter lab by running:

```
(myenv) (myenv) C:\Users\viet\spam_detection>jupyter lab
```

5.Build and Train the SMS Detection Model

Now that your environment is ready, you’re going to download the SMS training data and build a simple machine learning model to classify the SMS messages.

We downloaded the spam dataset for this project from here:

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

The datasets contain 5574 messages with respective labels of spam and ham (legitimate), With this data, we will train a machine learning model that can correctly classify SMS as ham or spam. These procedures will be carried out in a Jupyter notebook, which from our file directory is named 'project_notebok'.

6. Exploratory Data Analysis (EDA)

Here, we will apply a variety of techniques to analyze the data and get a better understanding of it.

a. Import Libraries and Data

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly_express as px
import wordcloud
import nltk
import warnings
warnings.filterwarnings('ignore')
```

The spam dataset located in the dataset directory named spam.csv can be imported as follows:

[2]:
```
df = pd.read_csv("C:/Users/viet/spam_detection/dataset/spam.csv", encoding='latin-1')
```

[3]:
```
df.head()
```

[3]:

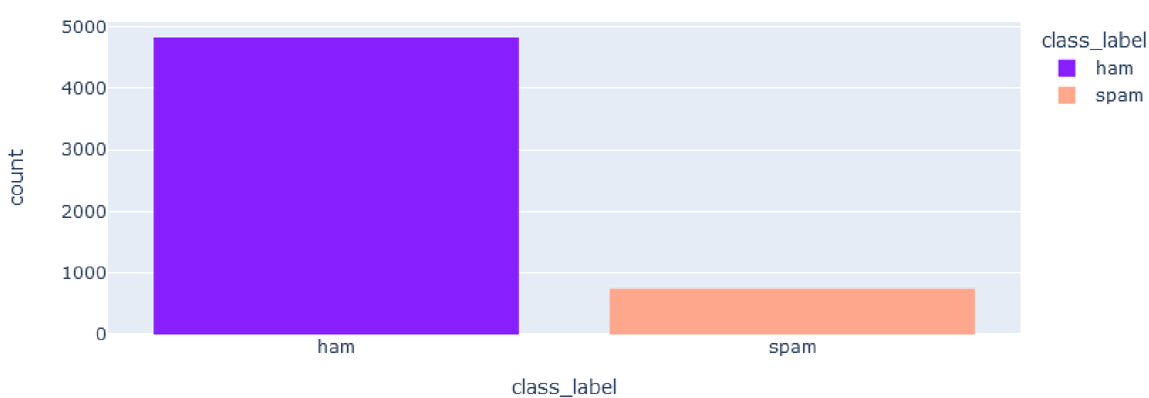
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

[4]:
```
df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace=True)
df.rename(columns = {'v1':'class_label','v2':'message'},inplace=True)
df.head()
```

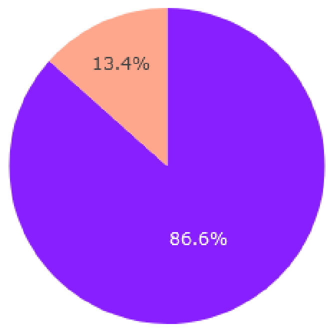
[4]:

	class_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...

[5]:
```
fig = px.histogram(df, x="class_label", color="class_label", color_discrete_sequence=["#871fff","#ffa78c"])
fig.show()
```



```
[6]: fig = px.pie(df.class_label.value_counts(),labels='index', values='class_label', color="class_label",
             color_discrete_sequence=["#871fff","#ffa78c"] )
fig.show()
```




```
[7]: df['length'] = df['message'].apply(len)
df.head()
```

[7]:

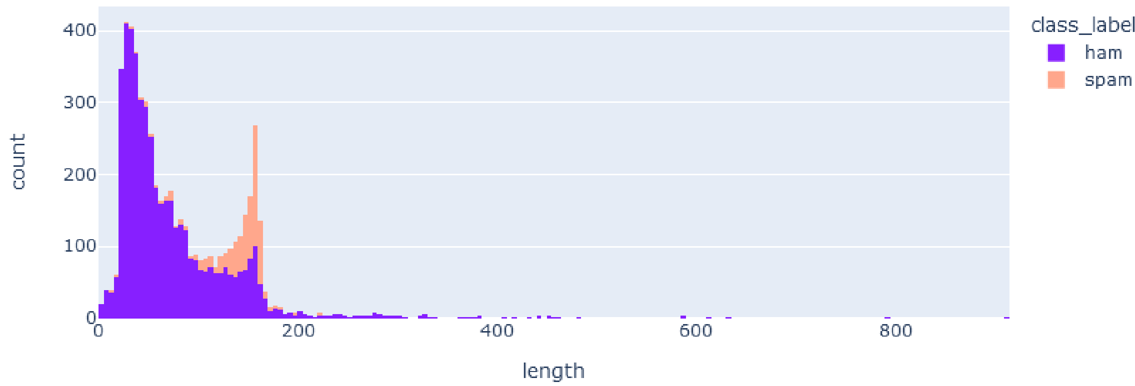
	class_label	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

```
[8]: df['length'] = df['message'].apply(len)
df.head()
```

[8]:

	class_label	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

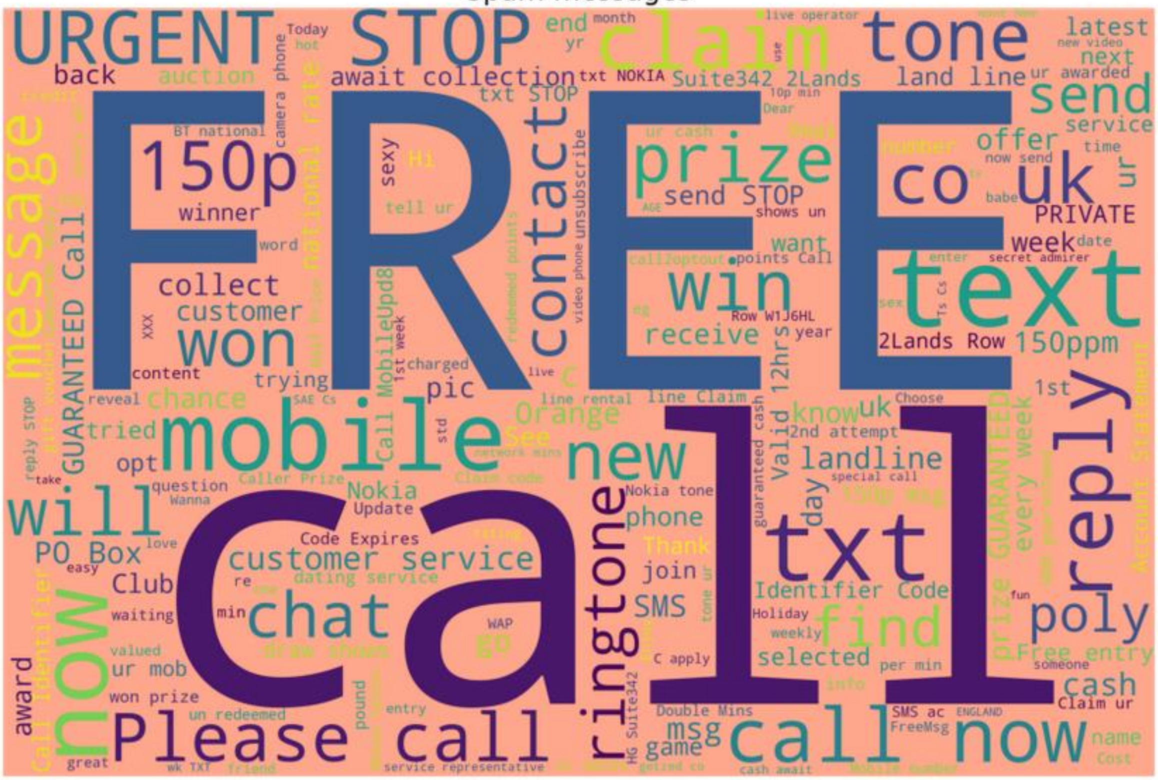
[9]:
```
fig = px.histogram(df, x="length", color="class_label", color_discrete_sequence=["#871fff","#ffa78c"] )
fig.show()
```



[10]:
```
data_ham  = df[df['class_label'] == "ham"].copy()
data_spam = df[df['class_label'] == "spam"].copy()

def show_wordcloud(df, title):
    text = ' '.join(df['message'].astype(str).tolist())
    stopwords = set(wordcloud.STOPWORDS)
    fig_wordcloud = wordcloud.WordCloud(stopwords=stopwords, background_color="#ffa78c",
                                        width = 3000, height = 2000).generate(text)
    plt.figure(figsize=(15,15), frameon=True)
    plt.imshow(fig_wordcloud)
    plt.axis('off')
    plt.title(title, fontsize=20)
    plt.show()
```

```
[11]: show_wordcloud(data_spam, "Spam messages")
```

Spam messages



```
[12]: show_wordcloud(data_ham, "ham messages")
```

ham messages


```
[13]: df['class_label'] = df['class_label'].map( {'spam': 1, 'ham': 0})
```

```
[14]: # Replace email address with 'emailaddress'
df['message'] = df['message'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$', 'emailaddress')

# Replace urls with 'webaddress'
df['message'] = df['message'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$', 'webaddress')

# Replace money symbol with 'money-symbol'
df['message'] = df['message'].str.replace(r'£|\$', 'money-symbol')

# Replace 10 digit phone number with 'phone-number'
df['message'] = df['message'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$', 'phone-number')

# Replace normal number with 'number'
df['message'] = df['message'].str.replace(r'\d+(\.\d+)?', 'number')

# remove punctuation
df['message'] = df['message'].str.replace(r'[^\w\d\s]', ' ')

# remove whitespace between terms with single space
df['message'] = df['message'].str.replace(r'\s+', ' ')

# remove leading and trailing whitespace
df['message'] = df['message'].str.replace(r'^\s+|\s*?$', ' ')

# change words to lower case
df['message'] = df['message'].str.lower()
```

```
[15]: from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
df['message'] = df['message'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))
```

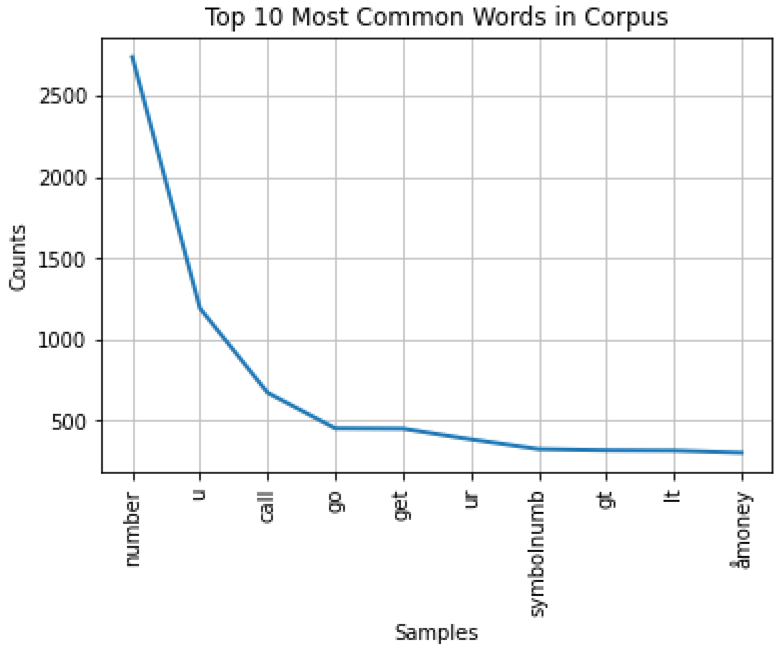
```
[16]: ss = nltk.SnowballStemmer("english")
df['message'] = df['message'].apply(lambda x: ' '.join(ss.stem(term) for term in x.split()))
```

```
[17]: import nltk

nltk.download('punkt')
sms_df = df['message']
from nltk.tokenize import word_tokenize

# creating a bag-of-words model
all_words = []
for sms in sms_df:
    words = word_tokenize(sms)
    for w in words:
        all_words.append(w)

all_words = nltk.FreqDist(all_words)
```

```
[18]: print('Number of words: {}'.format(len(all_words)))

Number of words: 6526
```

[19]:

```
all_words.plot(10, title='Top 10 Most Common Words in Corpus');
```



[23]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_model = TfidfVectorizer()
tfidf_vec=tfidf_model.fit_transform(sms_df)
import pickle
#serializing our model to a file called model.pkl
pickle.dump(tfidf_model, open("C:/Users/viet/spam_detection/model/tfidf_model.pkl","wb"))
tfidf_data=pd.DataFrame(tfidf_vec.toarray())
tfidf_data.head()
```

[23]:

	0	1	2	3	4	5	6	7	8	9	...	6496	6497	6498	6499	6500	6501	6502	6503	6504	6505
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	0.0	0.0	0.0
1	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	0.0	0.0
2	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	0.0	0.0
3	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	0.0	0.0
4	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	0.0	0.0

5 rows × 6506 columns

```
[24]: ### Separating Columns
df_train = tfidf_data.iloc[:4457]
df_test = tfidf_data.iloc[4457:]

target = df['class_label']
df_train['class_label'] = target

Y = df_train['class_label']
X = df_train.drop('class_label',axis=1)

# splitting training data into train and validation using sklearn
from sklearn import model_selection
X_train,X_test,y_train,y_test = model_selection.train_test_split(X,Y,test_size=.2, random_state=42)
```

```
[25]: import lightgbm as lgb
from sklearn.metrics import f1_score

def train_and_test(model, model_name):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    print(f'F1 score is: {f1_score(pred, y_test)}')

for depth in [1,2,3,4,5,6,7,8,9,10]:
    lgbmodel = lgb.LGBMClassifier(max_depth=depth, n_estimators=200, num_leaves=40)
    print(f"Max Depth {depth}")
    print(" ")
    print(" ")
    train_and_test(lgbmodel, "Light GBM")
```

```
Max Depth 1


F1 score is: 0.8870292887029289
Max Depth 2


F1 score is: 0.9236947791164659
Max Depth 3


F1 score is: 0.9149797570850203
Max Depth 4


F1 score is: 0.912
Max Depth 5


F1 score is: 0.9133858267716536
Max Depth 6


F1 score is: 0.9098039215686274
Max Depth 7


F1 score is: 0.9133858267716536
Max Depth 8


F1 score is: 0.9169960474308301
Max Depth 9


F1 score is: 0.9206349206349207
Max Depth 10


F1 score is: 0.9206349206349207
```

[28]:

```python
from sklearn.model_selection import RandomizedSearchCV
lgbmodel_bst = lgb.LGBMClassifier(max_depth=6, n_estimators=200, num_leaves=40)
param_grid = {
    'num_leaves': list(range(8, 92, 4)),
    'min_data_in_leaf': [10, 20, 40, 60, 100],
    'max_depth': [3, 4, 5, 6, 8, 12, 16, -1],
    'learning_rate': [0.1, 0.05, 0.01, 0.005],
    'bagging_freq': [3, 4, 5, 6, 7],
    'bagging_fraction': np.linspace(0.6, 0.95, 10),
    'reg_alpha': np.linspace(0.1, 0.95, 10),
    'reg_lambda': np.linspace(0.1, 0.95, 10),
    "min_split_gain": [0.0, 0.1, 0.01],
    "min_child_weight": [0.001, 0.01, 0.1, 0.001],
    "min_child_samples": [20, 30, 25],
    "subsample": [1.0, 0.5, 0.8],

}
model = RandomizedSearchCV(lgbmodel_bst, param_grid, random_state=1)
search = model.fit(X_train, y_train)
search.best_params_
```

[28]:

```
{'verbose': -1,
 'subsample': 0.5,
 'reg_lambda': 0.47777777777777775,
 'reg_alpha': 0.5722222222222222,
 'num_leaves': 88,
 'min_split_gain': 0.01,
 'min_data_in_leaf': 10,
 'min_child_weight': 0.01,
 'min_child_samples': 30,
 'max_depth': 3,
 'learning_rate': 0.1,
 'bagging_freq': 3,
 'bagging_fraction': 0.6}
```

```
[29]: best_model = lgb.LGBMClassifier(subsample=0.5,
                                      reg_lambda= 0.47777777777777775,
                                      reg_alpha= 0.5722222222222222,
                                      num_leaves= 88,
                                      min_split_gain= 0.01,
                                      min_data_in_leaf= 10,
                                      min_child_weight= 0.01,
                                      min_child_samples= 30,
                                      max_depth= 3,
                                      learning_rate= 0.1,
                                      bagging_freq= 3,
                                      bagging_fraction= 0.6,
                                      random_state=1)
      best_model.fit(X_train,y_train)
```

```
[29]: LGBMClassifier(bagging_fraction=0.6, bagging_freq=3, max_depth=3,
                     min_child_samples=30, min_child_weight=0.01, min_data_in_leaf=10,
                     min_split_gain=0.01, num_leaves=88, random_state=1,
                     reg_alpha=0.5722222222222222, reg_lambda=0.47777777777777775,
                     subsample=0.5)
```

```
[1]: prediction = best_model.predict(X_test)
     print(f'F1 score is: {f1_score(prediction, y_test)}')
```

```
F1 score is: 0.891566265060241
```

```
[32]: best_model.fit(tfidf_data, target)
      pickle.dump(best_model, open("C:/Users/viet/spam_detection/model/spam_model.pkl","wb"))
```