AI for Cyber Security Spam email detection

Spam Email detection

Dataset: https://www.kaggle.com/datasets/venky73/spam-mails-dataset/data

Machine learning provides a powerful way to combat spam emails. Here's the basic process:

- **Data Preparation:** Collect a labeled dataset of emails (spam and non-spam). Text data is cleaned and transformed into numerical features (e.g., word counts, presence of certain phrases).
- **Model Selection:** Choose a machine learning algorithm like Naive Bayes, Support Vector Machines (SVMs), or Random Forests.
- **Training:** Train the model on the prepared dataset, allowing it to learn patterns that distinguish spam from legitimate emails. Prediction: The trained model can now classify new, unseen emails as spam or not spam.

It discusses Enron emails and the data was collected from Enron1 folder.

Link: https://www2.aueb.gr/users/ion/data/enron-spam/

The dataset contains two folders of emails, spam and ham, each containing 517 emails.

The emails are labelled as

- spam
- ham

Importing Required Liberaries

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

Data Importing

```
df = pd.read_csv('spam_ham_dataset.csv')
df.head()
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
print(df['text'][0])
print(f"\nLabel: { df['label'][0] }")
```

```
Subject: enron methanol ; meter # : 988291
this is a follow up to the note i gave you on monday , 4 / 3 / 00 { preliminary
flow data provided by daren } .
please override pop ' s daily volume { presently zero } to reflect daily
activity you can obtain from gas control .
this change is needed asap for economics purposes .

Label: ham
```

```
print(df['text'][3])
print(f"\nLabel: { df['label'][3] }")
```

```
Subject: photoshop , windows , office . cheap . main trending
abasements darer prudently fortuitous undergone
```

```
lighthearted charm orinoco taster
railroad affluent pornographic cuvier
irvin parkhouse blameworthy chlorophyll
robed diagrammatic fogarty clears bayda
inconveniencing managing represented smartness hashish
academies shareholders unload badness
danielson pure caffein
spaniard chargeable levin


Label: spam
```

Data Preprocessing

```
df = df.rename(columns={df.columns[0]: 'word_count'})
```

```
df.head()
```

	word_count	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
df.sample(3)
```

	word_count	label	text	label_num
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
4135	1995	ham	Subject: re : occidental battleground meter 98...	0
4422	2977	ham	Subject: re : april 2001 spot purchases\r\nvan...	0

analyzing the word count of ham messages

```
df[df['label_num']==0].describe()['word_count']
```

```
count    3672.000000
mean     1835.500000
std      1060.159422
min         0.000000
25%       917.750000
50%      1835.500000
75%      2753.250000
max      3671.000000
Name: word_count, dtype: float64
```

analyzing word count of spam messages

```
df[df['label_num']==1].describe()['word_count']
```

```
count    1499.00000
mean     4421.00000
std       432.86834
min      3672.00000
25%      4046.50000
50%      4421.00000
75%      4795.50000
max      5170.00000
Name: word_count, dtype: float64
```

ML Application

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfid = TfidfVectorizer(max_features=3000)
```

```
X = tfid.fit_transform(df['text'])
```

Data Splitting

```
X.shape
```

```
(5171, 3000)
```

```
y = df['label_num']
```

```
y.shape
```

```
(5171,)
```

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

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=47)
```

```
X_train.shape, X_test.shape, y_train.shape ,y_test.shape
```

```
((4136, 3000), (1035, 3000), (4136,), (1035,))
```

Model Creation

```
from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression(C=1,solver='liblinear',penalty='l2', max_iter=50)
lr.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression(C=1, max_iter=50, solver='liblinear')
```

Model evaluation

```
y_pred = lr.predict(X_test)
```

```
from sklearn.metrics import r2_score, accuracy_score
```

```
print(r2_score(y_test,y_pred))
print(accuracy_score(y_test,y_pred))
```

```
0.940982484817958
0.9884057971014493
```