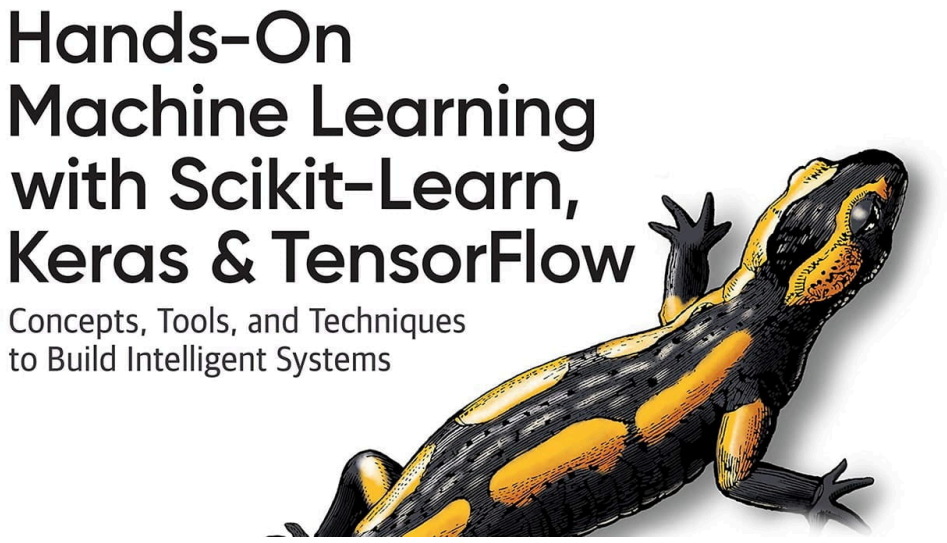
به نام خالق هستی



تمرین اول کارآموزی بامداد

بهار ۱۴۰۴

کارآموز

محمد امین شهابی

خرداد ۱۴۰۴

عنوان تمرین : تشخیص اسپم در ایمیل با استفاده از الگوریتمهای یادگیری ماشین

۱.پیش پردازش داده ها:

Deleting the bad format inputs and return every words in the document.

```python
def tokenize_email(text):
    words = re.findall(r'[a-zA-Z0-9]+', text)
    return [word.lower() for word in words]
```

Proccessing the .CSV input file.

+ Code + Markdown

```python
input_path = './data/emails.csv'
output_path = './data/emails_tokenized.csv'

with open(input_path, 'r', encoding='utf-8') as infile, \
     open(output_path, 'w', encoding='utf-8', newline='') as outfile:

    reader = csv.reader(infile)
    writer = csv.writer(outfile)

    writer.writerow(['tokens', 'spam'])

    next(reader, None)

    for row in reader:
        if len(row) < 2:
            continue

        text, label = row[0], row[1]
        tokens = tokenize_email(text)

        writer.writerow([tokens, label])

print("All done and saved in ",output_path)
```

توی این قسمت ما اومدیم با استفاده از **regex** کلماتی که شامل حروف انگلیسی و اعداد بودن رو از هم جدا کردیم و هر کدوم رو به یه توکن تبدیل کردیم(اگر لازم باشه ایمیل ها به زبان های دیگه ای جز انگلیسی باشن باید همین جا مشخص بشه.) و همه ایمیل های **Tokenize** شده رو داخل فایلی به نام `emails_tokenized.csv` در پوشه `data/.` ذخیره کردیم.

گام بعدی استخراج ویژگی های عددیست که به دو روش **Count Vectorization** و **Binary Vectorization** انجام میشن:

```python
tokenized_path = './data/emails_tokenized.csv'
df = pd.read_csv(tokenized_path)
print(f"Datas Before Converting :\n{df['tokens']}")
df['tokens'] = df['tokens'].apply(eval)
print(f"\nDatas After Converting :\n{df['tokens']}")
```

✓ 1.2s Python

```
Datas Before Converting :
0       ['subject', 'naturally', 'irresistible', 'your...
1       ['subject', 'the', 'stock', 'trading', 'gunsli...
2       ['subject', 'unbelievable', 'new', 'homes', 'm...
3       ['subject', '4', 'color', 'printing', 'special...
4       ['subject', 'do', 'not', 'have', 'money', 'get...
                              ...
5723    ['subject', 're', 'research', 'and', 'developm...
5724    ['subject', 're', 'receipts', 'from', 'visit',...
5725    ['subject', 're', 'enron', 'case', 'study', 'u...
5726    ['subject', 're', 'interest', 'david', 'please...
5727    ['subject', 'news', 'aurora', '5', '2', 'updat...
Name: tokens, Length: 5728, dtype: object

Datas After Converting :
0       [subject, naturally, irresistible, your, corpo...
1       [subject, the, stock, trading, gunslinger, fan...
2       [subject, unbelievable, new, homes, made, easy...
3       [subject, 4, color, printing, special, request...
4       [subject, do, not, have, money, get, software,...
                              ...
5723    [subject, re, research, and, development, char...
5724    [subject, re, receipts, from, visit, jim, than...
5725    [subject, re, enron, case, study, update, wow,...
5726    [subject, re, interest, david, please, call, s...
5727    [subject, news, aurora, 5, 2, update, aurora, ...
Name: tokens, Length: 5728, dtype: object
```

اولین قدم این بود که توکن هارو از حالت رشته هایی به شکل لیست در بیاریم و به لیست عادی تبدیل کنیم تا پایتون بفهمتشون !

سپس Vectorize میکنیم به دو صورت تعداد و وجود و عدم جود و ذخیرشون میکنیم :

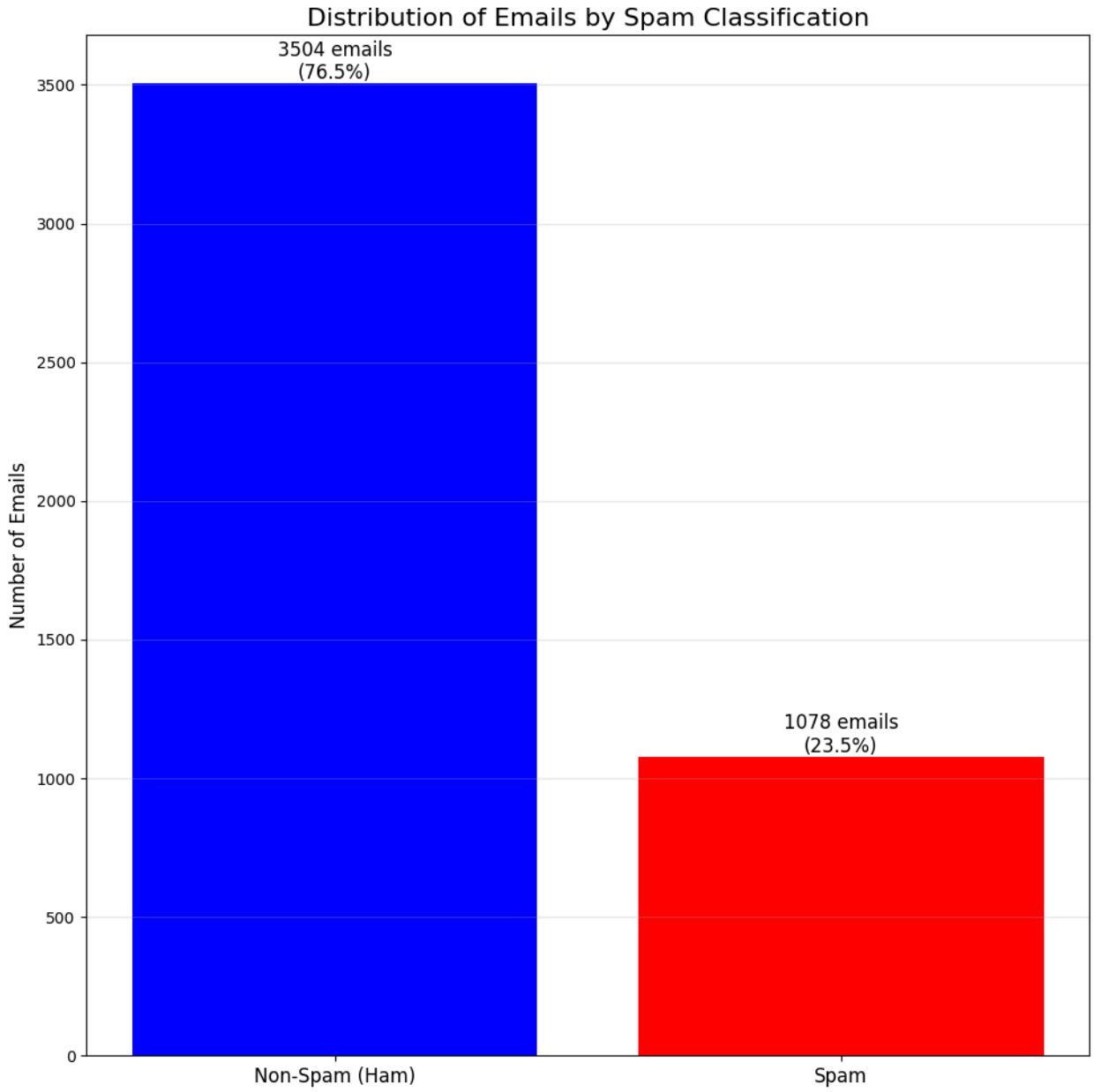
```
#1 Count Vectorization

count_vectorizer = CountVectorizer(
    tokenizer=lambda x: x,  # استفاده از توکن‌های از پیش پردازش شده
    preprocessor=lambda x: x,  # عدم پیش‌پردازش اضافی
    binary=False,  # حالت شمارشی
    min_df=2  # نادیده گرفتن کلماتی که کمتر از ۲ بار ظاهر شده‌اند
)
X_count = count_vectorizer.fit_transform(df['tokens'])
Python

#2 Binarry Vectorization

binary_vectorizer = CountVectorizer(
    tokenizer=lambda x: x,
    preprocessor=lambda x: x,
    binary=True,
    min_df=2
)
X_binary = binary_vectorizer.fit_transform(df['tokens'])
Python
```

گام بعد میشه آموزش مدل ولی داده های ما آماده این کار نیستن !

اول از همه میایم داده هامون رو به دو بخش test و train تقسیم میکنیم. ۲۰٪ از کل داده های اصلیمون رو اختصاص میدیم به test set و مدل در زمان یادگیری اون رو اصلا نمیبینه !

```
def shuffle_and_split_data(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]
✓ 0.0s
Python

train_set, test_set = shuffle_and_split_data(df, 0.2)
print(len(train_set),len(test_set))
✓ 0.0s
Python

4583 1145

print(df.applymap)
Python

np.random.seed(42)
✓ 0.0s
Python

from zlib import crc32
✓ 0.0s
Python

def is_id_in_test_set(identifier, test_ratio):
    return crc32(np.int64(identifier)) < test_ratio * 2**32

def split_data_with_id_hash(data, test_ratio, id_column):
    ids = data[id_column]
    in_test_set = ids.apply(lambda id_: is_id_in_test_set(id_, test_ratio))
    return data.loc[~in_test_set], data.loc[in_test_set]
✓ 0.0s
Python

emails_with_id = df.reset_index()  # adds an `index` column
train_set, test_set = split_data_with_id_hash(emails_with_id, 0.2, "index")
✓ 0.0s
Python

+ Code  + Markdown

df['index'] = df.index
✓ 0.0s
Python

emails_with_id["id"] = df['index'] * 1000 + df['spam']
train_set, test_set = split_data_with_id_hash(emails_with_id, 0.2, "id")
✓ 0.0s
Python

from sklearn.model_selection import train_test_split
✓ 0.0s
Python

train_set, test_set = train_test_split(df, test_size=0.2, random_state=42)
✓ 0.0s
Python
```

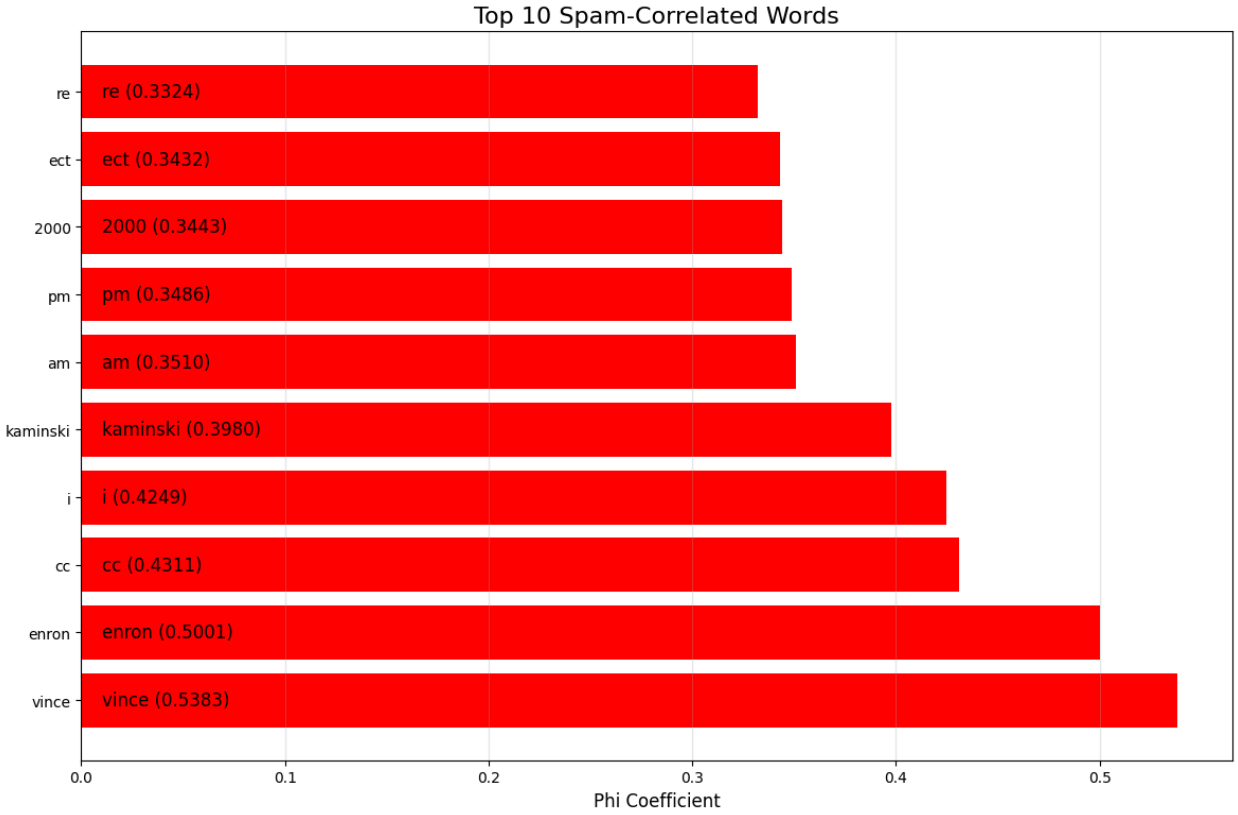
الان یه Train Set داریم که باید مدل رو باهاش آموزش بدیم :



تصویر بالا نشون میده که چند درصد از Train Set مختص به ایمیل های Spam و چند درصد مختص به Ham هاست !

الان دیگه آماده ایم؟؟ البته که نه! برای افزایش efficiency و performance مدل م چند گام دیگه رو هم بر میداریم که از سر باز نکرده باشیم .

گام اول پیدا کردن Correlation ها:



تصویر بالا نمودار phi confidence تمام لغات هستش. هر چقدر که یک لغت بیشتر توی ایمیل های Spam اومده باشه, phi confidence اون هم بیشتر میشه !

استفاده از Correlation ها میتونه کمک بزرگی بکنه بهمون ولی عیب های خاص خودش رو هم داره!

مزایای استفاده از Correlation ها :

کاهش Dimensionality حدودا ۹۰-۹۵٪

افزایش سرعت یادگیری و generalization (تعمیم)

معایب استفاده Correlation ها :

.وجود بیش از ۲۰K ویژگی برای کمتر از ۶K ایمیل! این به خودیه خود باعث میشه standard correlation بسیار noisy بشه !

راه حلی که من پیدا کردم استفاده از phi confidence بود

.بالانس نبودن میزان داده ها(Ham = 76%, Spam = 23%).

Phi Confidence formula :

$$\phi = \frac{(ad-bc)}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}$$

که در آن:

- a: تعداد مواردی که در هر دو متغیر مثبت هستند.
- b: تعداد مواردی که در متغیر اول مثبت و در متغیر دوم منفی هستند.
- c: تعداد مواردی که در متغیر اول منفی و در متغیر دوم مثبت هستند.
- d: تعداد مواردی که در هر دو متغیر منفی هستند.

به طور کلی، **Phi Coefficient** برای تحلیل داده‌های باینری و بررسی ارتباط بین دو متغیر دو حالته بسیار مفید است و می‌تواند در شرایطی که داده‌ها دارای نویز هستند، نتایج بهتری ارائه دهد.

Top 10 spam indicators:

	word	phi
17297	vince	0.538304
6419	enron	0.500109
3712	cc	0.431096
8497	i	0.424914
9503	kaminski	0.397970
2070	am	0.351010
12675	pm	0.348578
396	2000	0.344282
6080	ect	0.343242
13551	re	0.332392

Top 10 ham indicators:

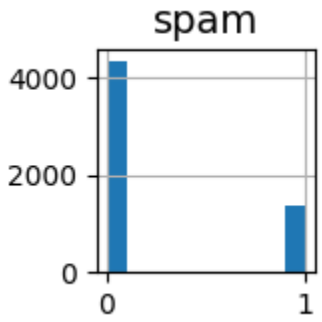
	word	phi
13838	reliably	0.0
13800	reinstated	0.0
13799	reinstate	0.0
13826	relaxing	0.0
3076	bl	0.0
3072	bjarne	0.0
3069	bitter	0.0
3067	bitmap	0.0
13844	relies	0.0
10272	lpharr	0.0

مقدار **1** نشان‌دهنده همبستگی کامل مثبت، **-1** نشان‌دهنده همبستگی کامل منفی و **0** نشان‌دهنده عدم همبستگی است.

فاز Data Cleaning :

به نظر من داده های فعلی خیلی نیاز نداشتن به این فاز . البته این بخاطر اینه که موقع Tokenize کردن ما فقط داده های درست رو Tokenize کردیم و همرو lowercase کردیم.

درگام بعدی هم دیدیم که هیچ سطری نیست که Spam یا Token رو null گذاشته باشه.



ما قبلا هم دیدم که داده هامون نامتعادلن. یعنی تعداد Ham ها خیلی از تعداد Spam ها بیشتره !!!

حالا چی کنیم این رو ؟؟

خب ما چندین راهه برای رفع این مشکل داریم :

1.Random Over Sampling 2.Random Under Sampling 3.SMOTE

۱.میاد از یه تعداد تصادفیی از داده های Spam کپی میگیره تا وقتی تعداد برابر شه.

این میتونه باعث OverFit شدن مدل روی اون داده ها بشه !!!

۲.به صورت تصادفی یه تعدادی از Ham هارو حذف میکنه تا وقتی داده ها برابر بشن.

خب سایز داده هامون خیلی کم و کوچیک میشه!!!

۳.میاد بر اساس Spam هایی که داره یکسری Spam دیگه به صورت مصنوعی میسازه.

ممکنه یکسری داده مصنوعی بی معنی بسازه و مدل روی اون OverFit بشه !!!

تمام نمودارها ارزیابی ها در فایل ارسالی داخل پوشه images می باشد!!

به نظر من بهترین گزینه استفاده از روش SMOTE بود و اون رو انتخاب کردم!

فاز Feature Scaling :

یکسری از مدل ها احتیاج دارن داده های Scale شده بگیرن وگرنه خروجی مد نظر رو نمیدن ! (مدل هایی مثل SVM,KNN, Logistic Regression)

فاز Training Models:

اومدیم همه داده های Train رو ذوی همه مدل های ذکر شده توی سند تست کردیم و این خروجی رو گرفتیم :

```
==================================================
BEST PERFORMING MODEL: Naive Bayes
AUC: 0.9990
Accuracy: 0.9939
Spam F1-score: 0.9872
==================================================


MODEL PERFORMANCE SUMMARY:
Model           Accuracy AUC      Spam F1  Train Time Test Time
Random Forest   0.9712  0.9939  0.9394  0.6727s      0.0270s
SVM             0.9520  0.9855  0.9020  6.5135s      0.8866s
KNN             0.4075  0.8212  0.4466  0.0037s      0.4620s
Decision Tree   0.9668  0.9619  0.9321  0.4911s      0.0047s
Naive Bayes     0.9939  0.9990  0.9872  0.0044s      0.0009s
```

که مشخص میکنه مدل Naive Bayes بهترین گزینست برای مدل ما !

اینم Matris Confusion این مدل میباشد که عملکرد خوب اون رو به خوبی میرسونه :