شناسایی ایمیل‌های جعلی

الف) پیش‌پردازش متن

مراحل پیش‌پردازش متن:

۱. حذف علامت‌گذاری، کلمات توقف، **URL** ها و **HTML**:
   - از re برای عملیات regex برای حذف URL ها و HTML استفاده کنید.
   - از string.punctuation برای حذف علامت های نگارشی و نوشتاری استفاده کنید. شامل کوچک کردن حروف و ... میشود.
   - از nltk.corpus.stopwords برای حذف کلمات توقف استفاده کنید.

پیش‌پردازش اختصارات:

- یک دیکشنری از اختصارات رایج و توسعه‌های آن‌ها ایجاد کنید.
- اختصارات را با توسعه‌های آن‌ها جایگزین کنید.

پیش‌پردازش یا حذف ایموجی‌ها و شکلک‌ها:

- از کتابخانه emoji برای حذف ایموجی‌ها استفاده کنید.
- از رجکس برای حذف شکلک‌های رایج استفاده کنید.

تأثیر بر عملکرد:

- پیش‌پردازش متن به کاهش نویز و حذف اطلاعات غیرضروری کمک می‌کند و بخشی ضروری است که باعث می‌شود مدل‌های یادگیری ماشین بهتر بتوانند الگوهای موجود در داده‌ها را شناسایی کنند. این مراحل باعث افزایش دقت مدل و کاهش خطاهای classification می‌شوند.

ب) تحلیل داده‌ها

کلمات پر تکرار در ایمیل‌های اسپم و غیر اسپم:

- از CountVectorizer از sklearn برای یافتن کلمات پر تکرار استفاده کنید.

- مجموعه داده‌ها را به ایمیل‌های اسپم و غیر اسپم تقسیم کنید.

کلماتی که احتمال اسپم بودن ایمیل را افزایش می‌دهند:

- کلمات رایج که احتمال اسپم بودن یک ایمیل را افزایش می‌دهند شامل اصطلاحات مرتبط با بازاریابی، فروش و درخواست‌های فوری مانند "offer" ،"free" ،"buy" و غیره هستند.

تأثیر بر عملکرد:

- تحلیل داده‌ها به شناسایی ویژگی‌های کلیدی که در طبقه‌بندی ایمیل‌ها به عنوان اسپم یا غیر اسپم مؤثر هستند، کمک می‌کند. این کلمات کلیدی می‌توانند به عنوان ویژگی‌های مهم در مدل‌های یادگیری ماشین استفاده شوند که باعث بهبود دقت مدل می‌شوند.

ج) متعادل‌سازی داده‌ها

روش‌های متعادل‌سازی مجموعه داده‌ها:

۱. تصادفی حذف کردن: حذف برخی نمونه‌ها از کلاس عمده.
۲. تصادفی افزوده کردن: تکرار برخی نمونه‌ها در کلاس اقلیت.
۳. **SMOTE** (روش مصنوعی افزوده‌سازی اقلیت) یا البته (**Synthetic Minority Over-sampling Technique**) **:** تولید نمونه‌های مصنوعی برای کلاس اقلیت.

استفاده از **SMOTE** برای متعادل‌سازی داده‌ها.

تأثیر بر عملکرد:

- متعادل‌سازی داده‌ها باعث کاهش مشکل عدم توازن کلاس‌ها می‌شود که می‌تواند منجر به بهبود دقت مدل و کاهش نرخ خطای دسته‌بندی کلاس اقلیت شود. استفاده از روش‌هایی مانند SMOTE به تولید داده‌های مصنوعی برای کلاس اقلیت کمک می‌کند که باعث افزایش تعداد نمونه‌های آموزشی و بهبود عملکرد مدل می‌شود.

د) توکن‌سازی

هدف از توکن‌سازی در روش‌های **NLP**:

- توکن‌سازی فرایند تقسیم متن به واحدهای کوچکتر به نام توکن‌ها (کلمات، جملات و غیره) است. این به تبدیل داده‌های متنی به فرمتی که می‌توان از آن برای آموزش مدل استفاده کرد، کمک می‌کند.

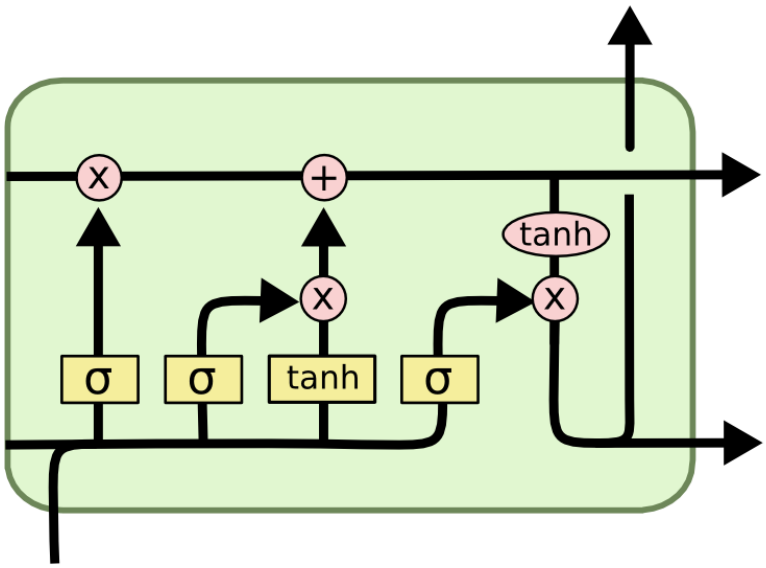
تأثیر بر عملکرد:

- توکن‌سازی باعث می‌شود متن به صورت عددی نمایه شود که برای مدل‌های یادگیری ماشین قابل استفاده باشد. این فرآیند به بهبود دقت مدل‌ها در تشخیص الگوهای موجود در داده‌های متنی کمک می‌کند و باعث افزایش کارایی مدل می‌شود.

ه) پیاده‌سازی مدل‌ها

LSTM یا (**Long Short-Term Memory**) / (حافظه بلند مدت کوتاه):

- توضیح: شبکه‌های **LSTM** نوعی شبکه عصبی بازگشتی (**RNN**) هستند که قادر به یادگیری وابستگی‌های طولانی مدت هستند. این شبکه‌ها برای داده‌های ترتیبی مناسب هستند و به منظور اجتناب از مشکل وابستگی طولانی طراحی شده‌اند. **LSTM** ها دارای معماری پیچیده‌تری نسبت به **RNN** های ساده هستند و شامل حالت سلولی و دروازه‌ها (ورودی، فراموشی، خروجی) هستند که جریان اطلاعات را تنظیم می‌کنند.



مثال پیاده‌سازی **LSTM**:

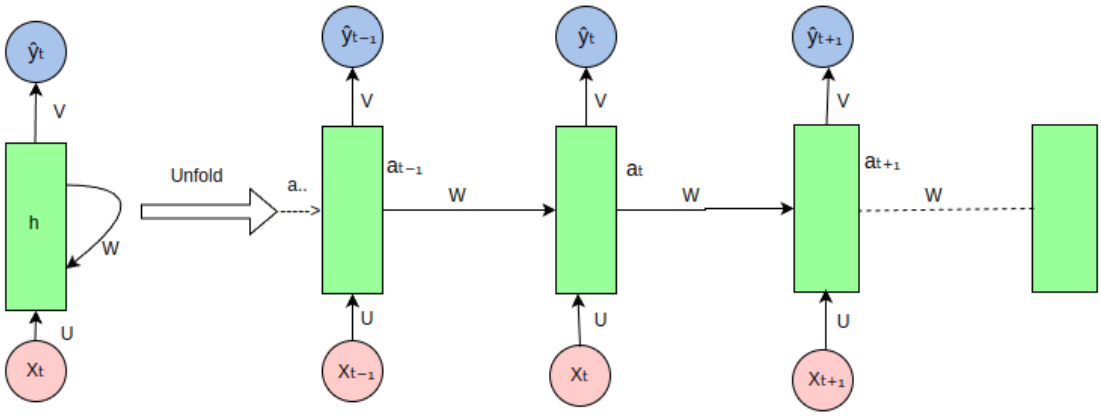
تأثیر بر عملکرد:

- مدل‌های **LSTM** به دلیل توانایی در یادگیری وابستگی‌های طولانی مدت و حفظ اطلاعات طولانی مدت، معمولاً عملکرد بهتری در پردازش داده‌های ترتیبی مانند متن دارند. این مدل‌ها می‌توانند الگوهای پیچیده‌تر را شناسایی کنند و دقت طبقه‌بندی را افزایش دهند.

```
35/35 [==============================] - 1s 29ms/step
              precision    recall  f1-score   support

         ham       0.99      0.94      0.97       966
        spam       0.72      0.94      0.82       149

    accuracy                           0.94      1115
   macro avg       0.86      0.94      0.89      1115
weighted avg       0.95      0.94      0.95      1115
```

RNN (شبکه عصبی بازگشتی):

- توضیح: **RNN** ها شبکه‌های عصبی هستند که دارای **loop** هایی هستند که اطلاعات را حفظ می‌کنند. این شبکه‌ها برای داده‌های **sequential** یا داده‌های سری زمانی **time series** استفاده می‌شوند. با این حال، **RNN** های استاندارد از مشکل ناپدید شدن گرادیان رنج می‌برند که باعث کاهش کارایی آن‌ها برای وابستگی‌های طولانی مدت می‌شود.



مثال پیاده‌سازی **RNN**:

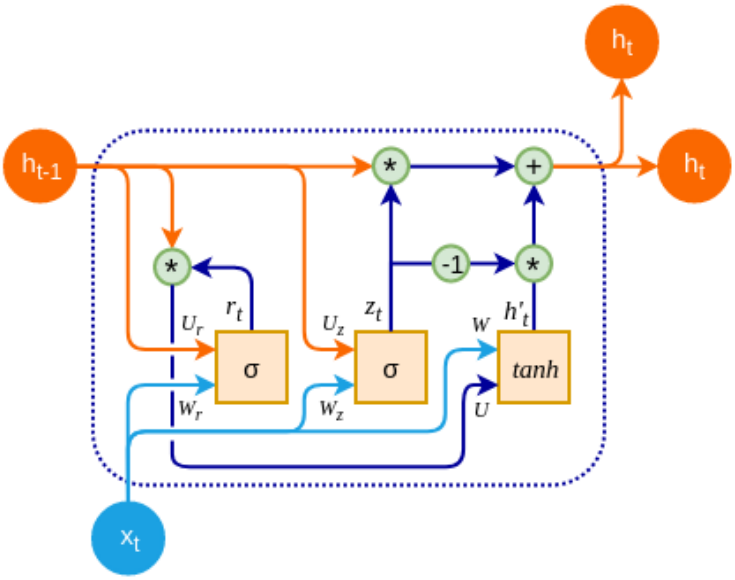
تأثیر بر عملکرد:

- مدل‌های **RNN** به دلیل سادگی و توانایی در پردازش داده‌های ترتیبی، برای مسائل ساده‌تر مناسب هستند. اما به دلیل مشکل ناپدید شدن گرادیان، معمولاً عملکرد آن‌ها در یادگیری وابستگی‌های طولانی مدت ضعیف‌تر است.

```
35/35 [==============================] - 1s 14ms/step
              precision    recall  f1-score   support

         ham       0.99      0.98      0.98       966
        spam       0.88      0.91      0.89       149

    accuracy                           0.97      1115
   macro avg       0.93      0.94      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```

GRU (واحد بازگشتی دروازه‌ای):

- توضیح: **GRU** ها نوعی از شبکه‌های **LSTM** هستند اما با معماری ساده‌تر. این شبکه‌ها دروازه ورودی و فراموشی را در یک دروازه به روز رسانی ترکیب می‌کنند که باعث افزایش کارایی محاسباتی آن‌ها می‌شود و همچنان مشکل ناپدید شدن گرادیان را حل می‌کنند.



مثال پیاده‌سازی **GRU**:

تأثیر بر عملکرد:

- مدل‌های **GRU** به دلیل سادگی بیشتر نسبت به **LSTM** ها و همچنان توانایی در یادگیری وابستگی‌های طولانی مدت، عملکرد مناسبی دارند. این مدل‌ها می‌توانند کارایی بالاتری داشته باشند و دقت مشابه یا بهتری نسبت به **LSTM** ها ارائه دهند.

```
35/35 [==============================] - 1s 31ms/step
              precision    recall  f1-score   support

         ham       0.99      0.96      0.97       966
        spam       0.77      0.96      0.85       149

    accuracy                           0.96      1115
   macro avg       0.88      0.96      0.91      1115
weighted avg       0.96      0.96      0.96      1115
```

مقایسه دقت و نمودارهای خطا:

- از **matplotlib** برای رسم نمودارهای دقت و خطا استفاده کنید.

گزارش دقت، یادآوری و **F1-Score**:

و) بهینه‌سازی مدل

آزمایش با انواع بهینه‌سازها و نرخ‌های یادگیری

- بهینه‌ساز و نرخ یادگیری را در مرحله کامپایل مدل تغییر دهید و عملکرد را ارزیابی کنید.

تأثیر بر عملکرد:

- انتخاب بهینه‌ساز مناسب و نرخ یادگیری بهینه می‌تواند تأثیر قابل توجهی بر عملکرد مدل داشته باشد. آزمایش با بهینه‌سازها و نرخ‌های یادگیری مختلف به یافتن بهترین ترکیب برای بهبود دقت و کاهش خطاهای مدل کمک می‌کند.

استفاده از کدی که در در بخش **Optimizing** شده شما را به این مرحله میبرد. در نهایت از همه **Optimizer** ها و **Learning Rate** ها استفاده کرده و بهترین **Combination** را منتخب میکند.

```
Training with optimizer rmsprop and learning rate 0.001...
Epoch 1/5
121/121 - 47s - loss: 0.3611 - accuracy: 0.8453 - val_loss: 0.1930 - val_accuracy: 0.9408 - 47s/epoch - 389ms/step
Epoch 2/5
121/121 - 41s - loss: 0.2209 - accuracy: 0.9155 - val_loss: 0.1103 - val_accuracy: 0.9713 - 41s/epoch - 336ms/step
Epoch 3/5
121/121 - 40s - loss: 0.1743 - accuracy: 0.9379 - val_loss: 0.1395 - val_accuracy: 0.9605 - 40s/epoch - 332ms/step
Epoch 4/5
121/121 - 40s - loss: 0.1466 - accuracy: 0.9495 - val_loss: 0.1178 - val_accuracy: 0.9677 - 40s/epoch - 335ms/step
Epoch 5/5
121/121 - 40s - loss: 0.1236 - accuracy: 0.9593 - val_loss: 0.1423 - val_accuracy: 0.9587 - 40s/epoch - 332ms/step
```

ذخیره کردن مدل:

مدل **Train** شده را در یک فایل با پسوند **h5** ذخیره کنید. دقت و **Precision** و **Recall** و **F1 Score** مدل را به راحتی با دستور **classification_report** که از کتابخانه **sklearn.metrics** خوانده بودیم بررسی میکنیم:

```
Test Set Evaluation:

Accuracy: 0.9587443946188341
Classification Report:
              precision    recall  f1-score   support

         ham       0.99      0.96      0.98       966
        spam       0.79      0.95      0.86       149

    accuracy                           0.96      1115
   macro avg       0.89      0.95      0.92      1115
weighted avg       0.96      0.96      0.96      1115
```

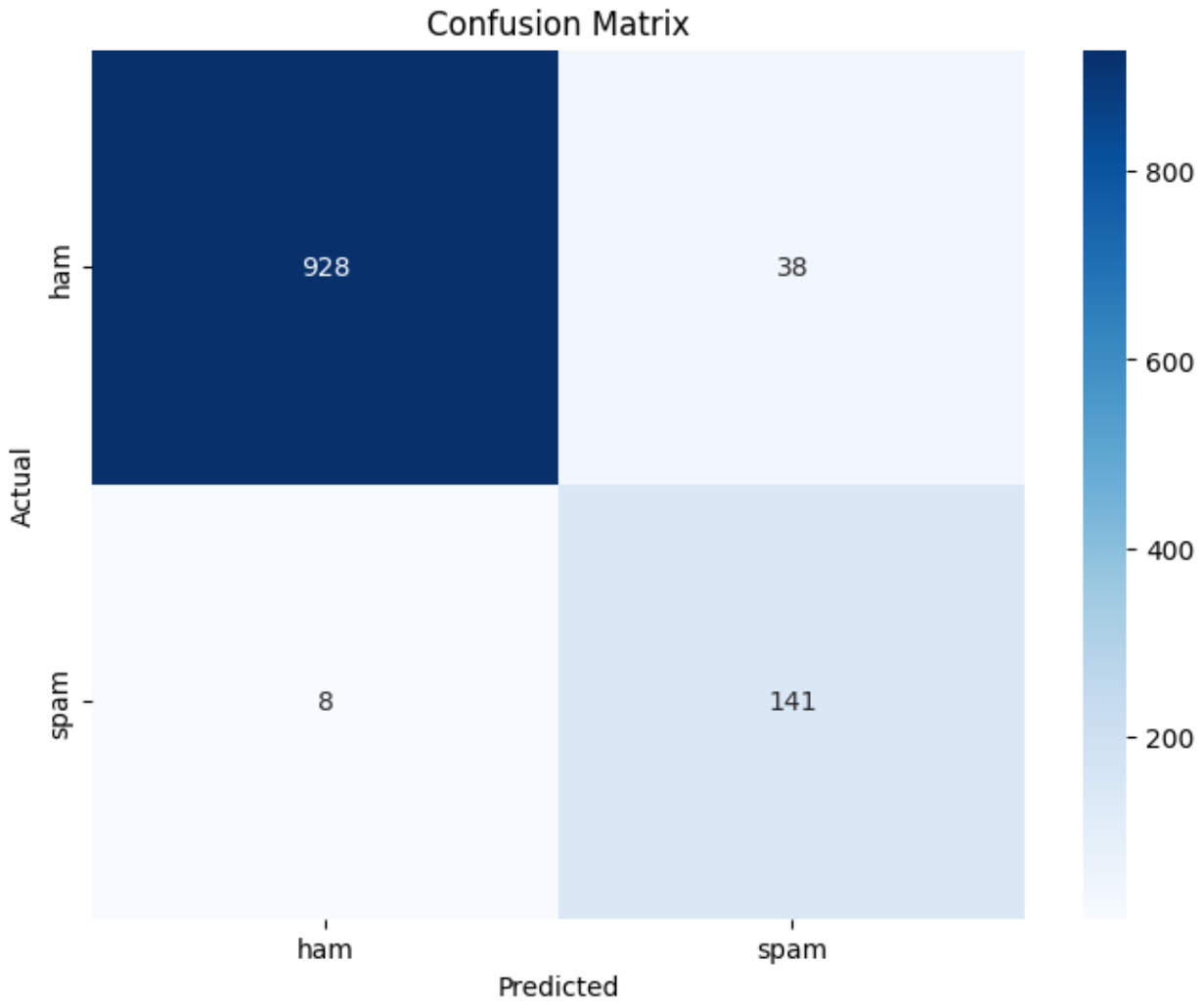
و) استفاده از مدل سیو شده

اگر مایل بودید مدل خاصی که در حال حاضر **Train** کرده اید را خوانده و از آن استفاده کنید میتوانید از کتابخانه **keras** استفاده کنید و لایه های مدل را ببینید:

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 100, 128)          640000

 spatial_dropout1d_9 (Spati  (None, 100, 128)          0
 alDropout1D)

 lstm_7 (LSTM)               (None, 100)               91600

 dense_9 (Dense)             (None, 1)                 101

=================================================================
Total params: 731701 (2.79 MB)
Trainable params: 731701 (2.79 MB)
Non-trainable params: 0 (0.00 Byte)
```

ماتریس سردرگمی **Confusion Matrix**

از **y_pred** و **y_test**


Confusion Matrix
ham
spam
Actual
928
38
8
141
ham
spam
Predicted
800
600
400
200