Trabajo Práctico 3 Detector de SPAM

Abril Noguera - Pablo Brahim - Fermin Rodriguez - Kevin Pennington

Uno de los problemas más comunes en la clasificación es la detección de correos electrónicos SPAM. Uno de los primeros modelos utilizados para abordar este problema fue el clasificador de Bayes ingenuo. La detección de SPAM es un problema persistente en el mundo digital, ya que los spammers continúan adaptando sus estrategias para eludir los filtros de correo no deseado. Además del clasificador de Bayes ingenuo, se han desarrollado y utilizado una variedad de técnicas más avanzadas en la detección de SPAM, que incluyen algoritmos de aprendizaje automático, redes neuronales y métodos basados en reglas.

En este trabajo práctico, utilizaremos un conjunto de datos que consta de 4601 observaciones de correos electrónicos, de los cuales 2788 son correos SPAM. Dado que el contenido de los correos electrónicos es un tipo de dato no estructurado, es necesario procesarlo de alguna manera. Para este conjunto de datos, ya se ha aplicado un procesamiento típico en el Procesamiento del Lenguaje Natural (NLP), que consiste en contar la frecuencia de palabras observadas en los correos. pueden utilizar técnicas más sofisticadas, como la extracción de características semánticas y el análisis de sentimientos, para mejorar la precisión de los modelos de detección de SPAM.

import pandas as pd import matplotlib.pyplot as plt from sklearn.model_selection import train_test_split

El procesamiento de lenguaje natural (NLP) desempeña un papel fundamental en la detección de SPAM, ya que permite analizar el contenido de los correos electrónicos y extraer características relevantes para la clasificación. Además de contar la frecuencia de palabras, se from sklearn.pipeline import Pipeline from sklearn.preprocessing import StandardScaler from sklearn.naive_bayes import MultinomialNB from sklearn.preprocessing import MinMaxScaler from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report from sklearn.metrics import roc_curve, roc_auc_score df = pd.read_csv("spambase.csv")

word_freq_make word_freq_address word_freq_all word_freq_our word_freq_over word_freq_internet word_freq_edu word_freq_edu word_freq_table word_freq_conference char_freq_; char_freq_(char_freq_! char_freq_! char_freq_! char_freq_edu word_freq_edu word_freq_address word_freq_address word_freq_stable word_freq_s 0 ... 0 640 640 320 0 0 0 0 778

In []: # Leemos el dataset In []: df.head() 940 ... 280 210 280 500 140 210 70 132 0 372 250 ... 2 0 710 0 1230 190 190 120 60 0 0 10 143 0 276 630 310 630 0 310 630 ... 137 0 137 0 0 630 0 310 630 310 630 ... 0 0 135 0 135 0

5 rows × 55 columns

Column

0 word_freq_make

2 word_freq_all word_freq_3d

word_freq_our

word_freq_over word_freq_remove

word_freq_order

word_freq_mail

word_freq_will

word_freq_people 13 word_freq_report

14 word_freq_addresses

16 word_freq_business

15 word_freq_free

17 word_freq_email

19 word_freq_credit

18 word_freq_you

20 word_freq_your 21 word_freq_font

22 word_freq_000

24 word_freq_hp

25 word_freq_hpl

27 word_freq_650

28 word_freq_lab

31 word_freq_857

33 word_freq_415

34 word_freq_85

38 word_freq_pm

40 word_freq_cs

44 word_freq_re

48 char_freq_;

49 char_freq_(

50 char_freq_[

51 char_freq_!

52 char_freq_\$

53 char_freq_#

dtypes: int64(55) memory usage: 1.9 MB

54 spam

In []: df.columns

45 word_freq_edu

46 word_freq_table

32 word_freq_data

36 word_freq_1999

37 word_freq_parts

39 word_freq_direct

41 word_freq_meeting

42 word_freq_original

43 word_freq_project

35 word_freq_technology 4601 non-null

47 word_freq_conference 4601 non-null

29 word_freq_labs

30 word_freq_telnet

23 word_freq_money

26 word_freq_george

word_freq_receive

word_freq_internet

<class 'pandas.core.frame.DataFrame'> RangeIndex: 4601 entries, 0 to 4600 Data columns (total 55 columns):

Non-Null Count Dtype

1 word_freq_address 4601 non-null int64

4601 non-null int64

4601 non-null int64

4601 non-null int64

4601 non-null int64

'word_freq_credit', 'word_freq_your', 'word_freq_font', 'word_freq_000', 'word_freq_money', 'word_freq_hp', 'word_freq_hpl', 'word_freq_george', 'word_freq_650', 'word_freq_lab', 'word_freq_labs', 'word_freq_telnet', 'word_freq_857', 'word_freq_data', 'word_freq_415', 'word_freq_85',

'word_freq_pm', 'word_freq_direct', 'word_freq_cs', 'word_freq_meeting',

1. ¿Cuáles son las 10 palabras más encontradas en correos con SPAM y en correos No SPAM? ¿Hay palabras en común? ¿Algunas llaman la atención?

Palabras más frecuentes en NO SPAM

Top 10 palabras en correos NO SPAM

data

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null 4601 non-null

4601 non-null 4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null 4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null 4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null

4601 non-null int64

Out[]: Index(['word_freq_make', 'word_freq_address', 'word_freq_all', 'word_freq_3d', 'word_freq_our', 'word_freq_over', 'word_freq_remove', 'word_freq_internet', 'word_freq_order', 'word_freq_mail', 'word_freq_receive', 'word_freq_will', 'word_freq_people', 'word_freq_report', 'word_freq_addresses', 'word_freq_free', 'word_freq_business', 'word_freq_email', 'word_freq_you',

'word_freq_technology', 'word_freq_1999', 'word_freq_parts',

'word_freq_original', 'word_freq_project', 'word_freq_re', 'word_freq_edu', 'word_freq_table', 'word_freq_conference', 'char_freq_;', 'char_freq_(', 'char_freq_[', 'char_freq_!',

'char_freq_\$', 'char_freq_#', 'spam'],

In []: # Obtener las columnas que contienen la frecuencia de palabras

word_columns = [col for col in df.columns if col.startswith('word_freq_')]

spam_word_freq_dict = {col.replace('word_freq_', ''): freq for col, freq in spam_word_freq.items()}

Palabras más frecuentes en SPAM

spam_top10 = dict(sorted(spam_word_freq_dict.items(), key=lambda item: item[1], reverse=True)[:10])

axes[0].barh(list(spam_top10.keys()), list(spam_top10.values()), color='tomato')

axes[1].barh(list(not_spam_top10.keys()), list(not_spam_top10.values()), color='seagreen')

not_spam_top10 = dict(sorted(not_spam_word_freq_dict.items(), key=lambda item: item[1], reverse=True)[:10])

Top 10 palabras en correos SPAM

make

not_spam_word_freq_dict = {col.replace('word_freq_', ''): freq for col, freq in not_spam_word_freq.items()}

spam_wc = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(spam_word_freq_dict)

not_spam_wc = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(not_spam_word_freq_dict)

eopl

The report remove mail business 85 mee 650 mee order all holographic forms or the first order all holographic forms of the first order forms or th

you

will

your

hpl

edu

address ·

meeting

En el contexto de clasificación de mails, el problema de SPAM suele ser desbalanceado, ya que en general sólo una fracción relativamente pequeña corresponde a SPAM, lo que puede dificultar que los modelos detecten correctamente la clase minoritaria.

0.0

0.5

1.0

1.5

2.0

Frecuencia promedio

2.5

3.0

3.5

1e6

3.5

Palabras en los mails no spam, como george, hp y hpl llaman la atención, en primera instancia. Pero deja de sorprender si se piensa que la base de datos fue creada por George Forman et. al. en HP Labs

En este caso la base de datos muestra una distribución relativamente equilibrada, con aproximadamente un 60% de correos NO SPAM y un 40% de SPAM, por lo que no se observa un desbalance severo.

4. Utilizando un clasificador de Regresión Logística, entrene con el conjunto de entrenamiento (en este caso, normalice los datos).

Para asegurar que la proporción de correos SPAM y NO SPAM se mantenga similar en los conjuntos de entrenamiento y prueba, se utiliza la opción stratify en el train_test_split, ya que de esta forma se respeta la distribución original de clases durante la partición de los datos.

5. Calcule la matriz de confusión del conjunto de evaluación para ambos modelos. ¿Qué tipo de error comete más cada modelo? ¿Cuál de los dos tipos de error crees que es más importante

Matriz de Confusión - Regresión Logística

Predicción

45

475

SPAM

- 700

- 600

- 500

- 400

- 300

- 200

- 100

4.0

1e6

3.0

2. Separe el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba (70% y 30% respectivamente).

george

Adjunto el link al repositorio con la resulución completa. Repositorio de Github

dtype='object')

Link al Repositorio

spam = df[df['spam'] == 1] not_spam = df[df['spam'] == 0]

Sumar frecuencias de palabras

from wordcloud import WordCloud

Crear nubes de palabras

spam_word_freq = spam[word_columns].sum()

Crear diccionarios de palabra:frecuencia

supongo, no se si se llama asi el paquete

fig, axs = plt.subplots(1, 2, figsize=(16, 8))

axs[0].imshow(spam_wc, interpolation='bilinear')

In []: # Ordenar y obtener las 10 palabras más frecuentes

axes[0].set_xlabel('Frecuencia promedio')

axes[1].set_xlabel('Frecuencia promedio')

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

axes[0].set_title('Top 10 palabras en correos SPAM')

axes[1].set_title('Top 10 palabras en correos NO SPAM')

Crear subplots

Gráfico para SPAM

axes[0].invert_yaxis()

Gráfico para NO SPAM

axes[1].invert_yaxis()

plt.tight_layout()

plt.show()

you

your

will

free

our

all

mail

email

business

remove

0.0

0.5

import matplotlib.pyplot as plt

plt.figure(figsize=(6, 5))

plt.ylabel('Porcentaje (%)')

plt.xlabel('Clase') plt.grid(axis='y') plt.ylim(0, 100) plt.tight_layout()

Graficar

plt.show()

100

80

20

In []: # Separar features y target

y = df['spam']

In []: nb_pipe = Pipeline([

X = df.drop(columns='spam')

('nb', MultinomialNB())

y_pred_nb = nb_pipe.predict(X_test)

0.97

In []: # Crear el pipeline: escalador + regresión logística

Entrenar el pipeline en el set de entrenamiento

print(classification_report(y_test, y_pred_logr))

0.92

0.92

0.92

precision recall f1-score support

0.93

0.92

0.91

0.95

0.91

0.92

0.91 0.87 0.89

('scaler', StandardScaler()),

print(classification_report(y_test, y_pred_nb))

0.77 0.96

precision recall f1-score support

0.89

0.86

0.87

0.82

0.87 0.89 0.87

nb_pipe.fit(X_train, y_train)

Pipeline

► MultinomialNB

In []: # esto es para ver que da

accuracy

pipe = Pipeline([

pipe.fit(X_train, y_train)

Pipeline

► StandardScaler

► LogisticRegression

In []: # Predecir sobre el conjunto de test

In []: # esto es para ver que da

accuracy

para este problema?

import matplotlib.pyplot as plt

axes[0].set_xlabel('Predicción')

axes[1].set_xlabel('Predicción')

axes[0].set_ylabel('Real')

axes[1].set_ylabel('Real')

Regresión Logística

plt.tight_layout()

plt.show()

Real

SPAM

from sklearn.metrics import confusion_matrix

cm_bayes = confusion_matrix(y_test, y_pred_nb)

Matriz de confusión para Regresión Logística cm_logr = confusion_matrix(y_test, y_pred_logr)

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

683

21

NO SPAM

anterior? Expanda su respuesta.

In []: # Precisión y Recall para Naive Bayes

print("Regresión Logística:")

Mostrar resultados print("Naive Bayes:")

email con información importante por considerarlo spam es mas grave.

precision_bayes = precision_score(y_test, y_pred_nb)

precision_logr = precision_score(y_test, y_pred_logr)

recall_bayes = recall_score(y_test, y_pred_nb)

Precisión y Recall para Regresión Logística

recall_logr = recall_score(y_test, y_pred_logr)

print(f" Precisión: {precision_bayes:.4f}") print(f" Recall: {recall_bayes:.4f}\n")

plt.ylabel('Tasa de Verdaderos Positivos (TPR)')

print(f"AUC Naive Bayes: {auc_bayes:.4f}")

print(f"AUC Regresión Logística: {auc_logr:.4f}")

0.2

0.4

Curvas ROC

plt.legend() plt.grid(True) plt.tight_layout()

plt.show()

1.0

0.8

Verdaderos Positivos (TPR)

de

0.2

0.0

0.0

AUC Naive Bayes: 0.8903

AUC Regresión Logística: 0.9691

Mostrar valores

axes[0].set_title('Matriz de Confusión - Naive Bayes', fontsize=14)

axes[1].set_title('Matriz de Confusión - Regresión Logística', fontsize=14)

Matriz de Confusión - Naive Bayes

Predicción

Matriz de confusión para Naive Bayes

macro avg weighted avg

In []: import seaborn as sns

Crear figura

Naive Bayes

y_pred_logr = pipe.predict(X_test)

macro avg

weighted avg

Porcentaje (%)

Conteo de clases en el dataset original

NO SPAM

Realizar el split 70% train, 30% test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y

1.0

In []: common_words = set(spam_top10.keys()) & set(not_spam_top10.keys()) print("\nPALABRAS EN COMÚN EN AMBOS TOP 10:", common_words)

PALABRAS EN COMÚN EN AMBOS TOP 10: {'your', 'you', 'will'}

class_counts = df['spam'].value_counts(normalize=True) * 100

plt.title('Distribución de clases en el conjunto de datos original')

Distribución de clases en el conjunto de datos original

Clase

1.5

2.0

Frecuencia promedio

plt.bar(class_counts.index.map({0: 'NO SPAM', 1: 'SPAM'}), class_counts.values, color=['seagreen', 'tomato'])

SPAM

3. Utilizando un clasificador de Bayes ingenuo, entrene con el conjunto de entrenamiento.

837

544

1381

1381

('logreg', LogisticRegression(random_state=42, max_iter=1000)) # Paso 2: ajustar el modelo

837

544

1381

1381

1381

sns.heatmap(cm_bayes, annot=True, fmt='d', cmap='Blues', xticklabels=['NO SPAM', 'SPAM'], yticklabels=['NO SPAM', 'SPAM'], ax=axes[0])

sns.heatmap(cm_logr, annot=True, fmt='d', cmap='Blues', xticklabels=['NO SPAM', 'SPAM'], yticklabels=['NO SPAM', 'SPAM'], ax=axes[1])

154

523

SPAM

- 600

- 500

400

- 300

- 200

- 100

SPAM

792

69

NO SPAM

Paso 1: normalizar los datos

('scaler', MinMaxScaler()), # min max porque necesitamos inputs no negativos

2.5

axs[1].imshow(not_spam_wc, interpolation='bilinear')

axs[0].set_title('Palabras más frecuentes en SPAM', fontsize=18)

axs[1].set_title('Palabras más frecuentes en NO SPAM', fontsize=18)

nternet

not_spam_word_freq = not_spam[word_columns].sum()

In []: # Separar SPAM y NO SPAM

In []: #%pip install wordcloud

Graficar

axs[0].axis('off')

axs[1].axis('off')

plt.tight_layout()

plt.show()

4601 non-null int64

In []: df.info()

print(f" Precisión: {precision_logr:.4f}") print(f" Recall: {recall_logr:.4f}") Naive Bayes: Precisión: 0.7725 Recall: 0.9614 Regresión Logística: Precisión: 0.9135 Recall: 0.8732 Naive Bayes prioriza capturar todo el spam a costa de perder algunos correos válidos (por eso el alto recall), mientras que con el modelo de Regresión Logística se está priorizando no bloquear correos que sean legítimos (dejar tener algunos falsos negativos). El mejor modelo

Naive Bayes comete falsos positivos, mientras que Regresión Logística comete falsos negativos (aunque la diferencia es prácticamente despreciable). Para un filtro de spam es mejor tener falsos negativos, dejando pasar mails spam y que el usuario los desestime. Perder un

6. Calcule la precisión y la recuperación de ambos modelos. Para cada métrica, ¿cuál es el mejor modelo? ¿Cómo se relacionan estas métricas con los tipos de errores analizados en el punto

en este caso y como se dijo en el punto anterior es el de Regresión Logística, y que el usuario haga algo de trabajo. 7. Obtenga la curva ROC y el AUC (Área Bajo la Curva ROC) de ambos modelos. In []: # Probabilidades de predicción (no solo 0/1) y_prob_nb = nb_pipe.predict_proba(X_test)[:, 1] y_prob_logr = pipe.predict_proba(X_test)[:, 1] # Curva ROC para Naive Bayes fpr_bayes, tpr_bayes, _ = roc_curve(y_test, y_prob_nb) auc_bayes = roc_auc_score(y_test, y_prob_nb) # Curva ROC para Regresión Logística fpr_logr, tpr_logr, _ = roc_curve(y_test, y_prob_logr) auc_logr = roc_auc_score(y_test, y_prob_logr) # Graficar ambas curvas ROC plt.figure(figsize=(8,6)) plt.plot(fpr_bayes, tpr_bayes, label=f'Naive Bayes (AUC = {auc_bayes:.4f})', color='tomato') plt.plot(fpr_logr, tpr_logr, label=f'Logistic Regression (AUC = {auc_logr:.4f})', color='seagreen') plt.plot([0,1], [0,1], 'k--', label='Random Classifier') plt.title('Curvas ROC') plt.xlabel('Tasa de Falsos Positivos (FPR)')

> Naive Bayes (AUC = 0.8903) Logistic Regression (AUC = 0.9691) --- Random Classifier 0.6 0.8 1.0 Tasa de Falsos Positivos (FPR)

La comparación de las curvas ROC muestra que el modelo de Regresión Logística presenta un desempeño superior al modelo de Naive Bayes. Esto se refleja en el valor del AUC, que alcanza 0.9691 para la regresión logística frente a 0.8903 para Naive Bayes. Por lo tanto,

podemos concluir que, en este caso, la Regresión Logística es el clasificador más robusto, logrando una mejor tasa de verdaderos positivos a lo largo de todos los posibles umbrales de decisión.

Link al Notebook

Se puede encontrar el trabajo completo en el siguiente link: Repositorio GitHub