UNIVERSIDAD DE CUNDINAMARCA

FACULTAD DE INGENIERÍA

Clasificación de Correos HAM-SPAM

801

Francisco Antonio Vargas

Juan Manuel Pastor

Estudiantes

Machine Learning

Alexander Espinosa

Docente

Facatativá

10/09/2025

Clasificación de Correos HAM-SPAM

Conceptos

Para abordar el tema de la clasificación de correos es necesario tener presentes diversas definiciones fundamentales que comprueban su utilidad y funcionamiento, lo primero es tener en claro que es la regresión logística, esta se emplea como algoritmo principal, prediciendo probabilidades para la clasificación binaria entre "ham" y "spam" mediante la función sigmoide.

Las features, consisten en variables predictivas que el modelo usa para aprender patrones, como lo son el número de enlaces o signos de exclamación, son vitales para determinar la calidad que afecta el rendimiento.

El pseudo-etiquetado es normalmente utilizado para crear etiquetado cuando no existen etiquetas previas, sin embargo, esta aproximación puede introducir sesgos si las reglas heurísticas no son precisas.

La curva ROC junto con su área bajo la curva, conocida como AUC, son indicadores esenciales que miden la habilidad del modelo para diferenciar entre diversas clases, donde un AUC que se aproxime a 1 sugiere un desempeño excelente.

La normalización de los datos ajusta las características para que valores más altos no dominen al modelo, lo que a su vez potencia la convergencia durante el proceso de entrenamiento. Por último, la colinealidad se refiere a las conexiones fuertes entre las características, que pueden disminuir la efectividad del modelo si no se controlan adecuadamente, resaltando la necesidad de examinar estas interrelaciones para mejorar el

análisis. Juntos, estos elementos ofrecen un fundamento sólido para comprender y perfeccionar la metodología del código.

Descripción del código

Dado el problema de clasificar la cantidad de correos ham o spam que le llegan a un usuario, se diseño un dataset especifico con mil instancias de correos, los cuales simularan el comportamiento habitual en el flujo de correos recibidos, usando regresión lineal.

El primer paso es la importación de las librerías necesarias, para luego cargar los datos desde el archivo .svc preparado anteriormente, este cuenta con las columnas id, subject, body, y email.

```
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, roc_curve,
auc
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Carga del dataset
df = pd.read_csv('emails_dataset.csv')
```

Para la correcta clasificación entre spam y ham se generan etiquetas en donde 0 corresponde a los correos legítimos o ham y el 1 a los correos no deseados o spam, para ello se revisará que el correo contenga alguna de las palabras clave como oferta, gratis, entre otras, utilizando la función label_email (), esto con el fin de crear un conjunto de entrenamiento inicial que separara aproximadamente la mitad de los datos entre ham y spam.

```
# Generación de pseudo-etiquetas
spam_keywords = ['oferta', 'gratis', 'premio', 'ganado', 'bloqueada',
'crÃ©dito', 'descuento', 'inversiÃ³n', 'promociÃ³n', 'viaje', 'dinero',
'trabajando', 'oportunidad', 'exclusiva', 'seleccionada', 'medicamentos',
'cancÃºn']
def label_email(row):
    text = (row['subject'] + ' ' + row['body']).lower()
    if any(keyword in text for keyword in spam_keywords):
        return 1
    return 0
df['label'] = df.apply(label_email, axis=1)
```

El próximo paso consiste en la extracción de los features, aquí se calculan los 10 features que se preestablecieron previamente, es decir, número de enlaces, mención de "oferta" o "gratis", signos de exclamación, remitente conocido, cantidad de palabras, imágenes, adjuntos, palabras en mayúsculas, signos monetarios, y emoticones, de esta manera se transforma el texto en variables numéricas que el modelo puede usar para aprender patrones de spam vs. ham.

```
# Extracción de features
def extract_features(row):
    text = row['body'] + ' ' + row['subject']
    num_links = len(re.findall(r'http[s]?://', text))
    mentions_offer_free = 1 if re.search(r'\boferta\b|\bgratis\b|
    \bpromociÃ³n\b', text.lower()) else 0
    num_exclamations = text.count('!')
    sender_known = 1 if any(domain in row['email'] for domain in ['gmail.com',
    'outlook.com', 'hotmail.com']) else 0
    num_words = len(text.split())
    num_images = 0
    has_attachment = 1 if 'adjunto' in text.lower() or 'adjuntamos' in text.
    lower() else 0
    all_caps_words = len([word for word in text.split() if word.isupper()])
    has_money_symbols = 1 if '$' in text or '%' in text else 0
    num_emoticons = len(re.findall(r'[:;][\)\(DPO]', text))
    return pd.Series([num_links, mentions_offer_free, num_exclamations,
    sender_known, num_words, num_images, has_attachment, all_caps_words,
    has_money_symbols, num_emoticons])

features = ['num_links', 'mentions_offer_free', 'num_exclamations',
'sender_known', 'num_words', 'num_images', 'has_attachment', 'all_caps_words',
'has_money_symbols', 'num_emoticons']
df[features] = df.apply(extract_features, axis=1)
```

Se realiza un mapeo con nombres, de las variables a su equivalente para manejar las descripciones de manera legible.

```
# Mapeo de nombres técnicos a descripciones legibles
feature_names = {
    'num_links': 'Número de enlaces',
    'mentions_offer_free': 'Mención de oferta o gratis',
    'num_exclamations': 'Cantidad de signos de exclamación',
    'sender_known': 'Remitente conocido',
    'num_words': 'Cantidad de palabras',
    'num_images': 'Cantidad de imágenes',
    'has_attachment': 'Archivos adjuntos',
    'all_caps_words': 'Palabras en mayúsculas',
    'has_money_symbols': 'Signos de $ o %',
    'num_emoticons': 'Uso de emoticones'
}
```

El siguiente paso para es dividir las features como X y las etiquetas como Y, también se separa el dataset para que el 80% se destine al entrenamiento y el 20% a las pruebas con el uso de train_test_split, para luego escalar los datos con starndardscaler (), esto prepara los datos para el modelo y asegura un entrenamiento robusto evitando así los posibles sesgos por magnitudes diferentes en las features.

```
# Seleccionar un subconjunto de features para el heatmap
selected_features = ['num_links', 'mentions_offer_free', 'num_exclamations',
'all_caps_words', 'has_money_symbols']
df_correlation = df[selected_features].rename(columns={f: feature_names[f] for
f in selected_features})

# Preparación de datos
X = df[features]  # Matriz de features
y = df['label']  # Etiquetas
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)  # Divide 80% train, 20% test
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Continuando con el entrenamiento del modelo se deberá instanciar y entrenar con la función logística por medio del logisticregresion, esto se hace para ajustar el modelo a los datos de entrenamiento para predecir etiquetas con base en los features. Luego sigue la evaluación del entrenamiento, lo que predice las etiquetas en el conjunto de prueba, y calcula la precisión para generar un reporte detallado, con el fin de evaluar el rendimiento del modelo. Ya con esto se procede a realizar la predicción en el dataset completo, mapeando predicciones numéricas (0, 1) a etiquetas legibles ("Ham", "Spam"), clasificando así todas las instancias.

.

```
# Entrenamiento del modelo
model = LogisticRegression(max_iter=1000)  # Instancia el modelo
model.fit(X_train_scaled, y_train)  # Entrena con datos escalados

# Evaluación del modelo
y_pred = model.predict(X_test_scaled)  # Predice en test
accuracy = accuracy_score(y_test, y_pred)  # Calcula precisión
report = classification_report(y_test, y_pred)  # Reporte detallado

# Predicción en el dataset completo
X_full_scaled = scaler.transform(X)  # Escala todo el dataset
df['predicted_label'] = model.predict(X_full_scaled)  # Predicciones
df['classification'] = df['predicted_label'].apply(lambda x: 'Spam' if x == 1
else 'Ham')  # Mapea a strings
```

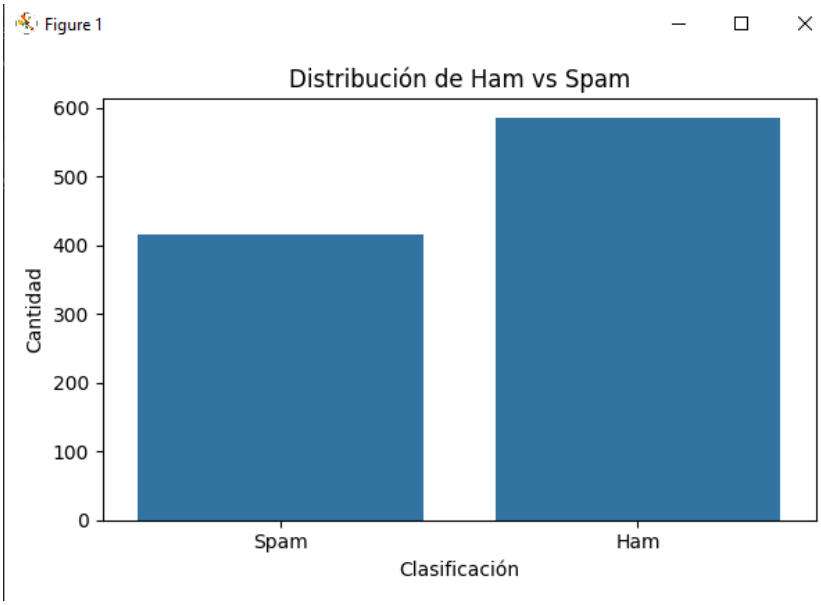
Para comprobar el comportamiento se realizaron graficas para la distribución de etiquetas entre ham y spam, la curva ROC, y por ultimo la importancia de las features, luego de esas graficas se imprime en consola los resultados, lo que muestra la precisión, el reporte de clasificación, las clasificaciones de los primeros 20 correos, y el conteo total de ham/spam, lo que facilita la interpretación visual de los resultados.

```
# Generación de gráficas
# 1. Distribución de etiquetas (ham/spam)
plt.figure(figsize=(6, 4))
sns.countplot(x='classification', data=df)
plt.title('Distribución de Ham vs Spam')
plt.xlabel('Clasificación')
plt.ylabel('Cantidad')
plt.tight_layout()
plt.show()

# 2. Curva ROC
y_prob = model.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.
2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC')
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()
```

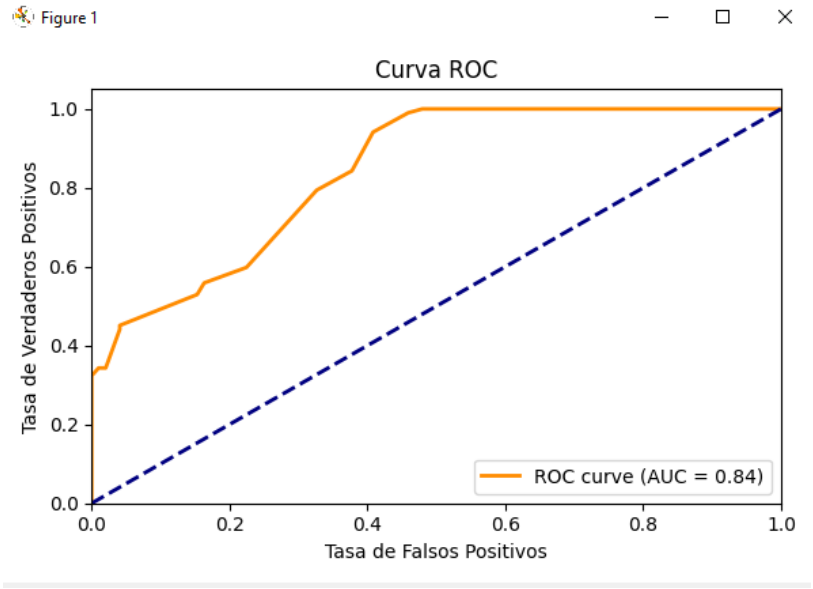
```
# 3. Importancia de las features (gráfico de puntos)
feature_importance = pd.DataFrame({'Feature': features, 'Coefficient': model.
coef_[0]})
feature_importance['Feature'] = feature_importance['Feature'].map
(feature_names)  # Renombrar features
feature_importance = feature_importance.sort_values(by='Coefficient',
ascending=False)
plt.figure(figsize=(8, 6))
plt.scatter(feature_importance['Coefficient'], feature_importance['Feature'],
s=100, c=feature_importance['Coefficient'], cmap='coolwarm')
plt.colorbar(label='Coeficiente')
plt.title('Importancia de las Features (Coeficientes del Modelo)')
plt.xlabel('Coeficiente')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()

# Impresión de resultados en terminal
print("Precisión en el conjunto de test:", accuracy)
print("Reporte de Clasificación:\n", report)
print("\nClasificaciones del Dataset Completo (primeros 20 para brevedad):\n")
print(df[['id', 'subject', 'classification']].head(20).to_string(index=False))
print("\nConteo Total de Ham y Spam (balance aproximado 50/50):")
print(df['classification'].value_counts().to_string())
```

Resultados

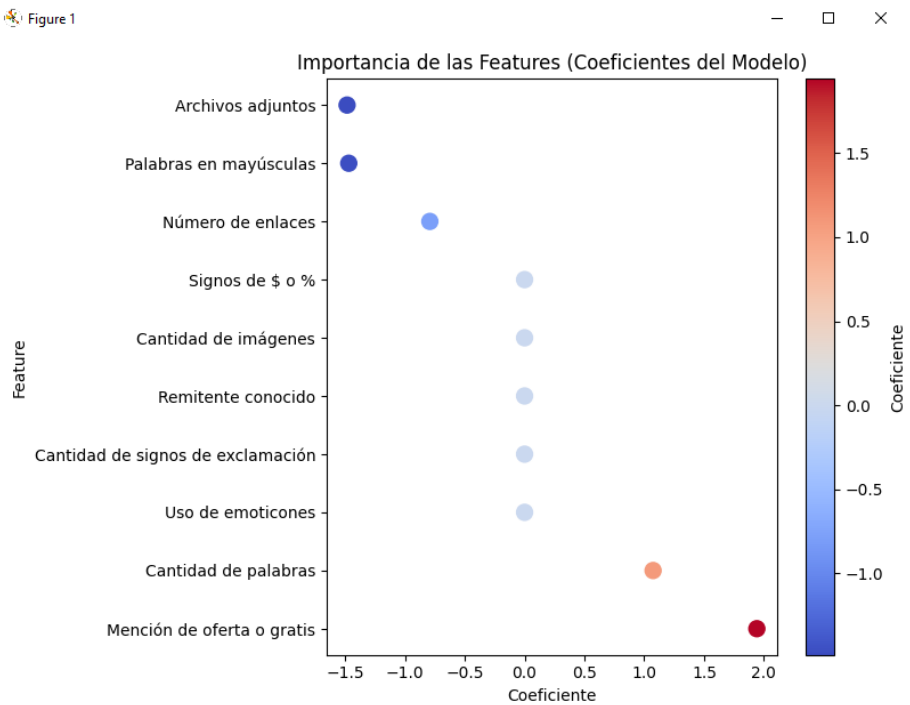
La grafica de distribución de ham VS spam muestra de manera simple ya que muestra la cantidad de correos clasificados como "Ham" (legítimos) y "Spam" (no deseados) en el dataset completo.



Esta gráfica traza la tasa de verdaderos positivos (sensibilidad) contra la tasa de falsos positivos a diferentes umbrales de clasificación. Incluye una línea diagonal de referencia y muestra el área bajo la curva (AUC), indicando la capacidad del modelo de regresión logística para distinguir entre "Ham" y "Spam", con un AUC más cercano a 1 reflejando mejor rendimiento.



La grafica de la importancia de los features usa puntos coloreados según el valor del coeficiente, lo que hace más evidente la magnitud y dirección (positiva/negativa) de la influencia, mejorando la comprensión de qué features impulsan la clasificación a "spam" o "ham".



A continuación se mostrara la salida en la terminal la cual muestra un reporte de la clasificación teniendo la clase ham (0) con una precisión de 0.66, un recall de 0.77 un f1-score 0.71 y un soporte de 98, además la clase spam (1) con una precisión de 0.73, un recall de 0.62 un f1-score 0.67 y un soporte de 102. Posteriormente muestra la clasificación de los primeros 20 correos. Y por ultimo se realiza un conteo de cuantos son los correos spam y ham, demostrando que hay 585 ham y 415 spam en total.

```
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.66      0.77      0.71        98
           1       0.73      0.62      0.67       102

    accuracy                           0.69       200
   macro avg       0.70      0.69      0.69       200
weighted avg       0.70      0.69      0.69       200


Clasificaciones del Dataset Completo (primeros 20 para brevedad):

 id                                subject classification
  1             Agenda semanal del proyecto           Spam
  2              Tu tarjeta serÃ¡ bloqueada            Ham
  3             Agenda semanal del proyecto           Ham
  4              Bienvenido a la plataforma           Ham
  5              ReuniÃ³n de equipo maÃ±ana            Ham
  6 Ãltima oportunidad: crÃ©dito inmediato           Ham
  7                    Has ganado un premio           Ham
  8        Factura del servicio de internet           Ham
  9           Cambio de contraseÃ±a exitoso           Ham
 10           Cambio de contraseÃ±a exitoso           Ham
 11              ReuniÃ³n de equipo maÃ±ana           Ham
 12        Factura del servicio de internet           Ham
 13               ActualizaciÃ³n de sistema           Spam
 14              ReuniÃ³n de equipo maÃ±ana            Ham
 15          Tu cuenta ha sido seleccionada           Spam
 16     Aprovecha este descuento increÃ ble            Ham
 17 Ãltima oportunidad: crÃ©dito inmediato           Ham
 18              Entrega de informe mensual           Spam
 19     PromociÃ³n especial en medicamentos           Spam
 20               ActualizaciÃ³n de sistema            Ham

Conteo Total de Ham y Spam (balance aproximado 50/50):
classification
Ham     585
Spam    415
```

LINK DE GITHUB: https://github.com/JuanMPastor/Clasificador-de-Correos.git