Introducción a la Inteligencia Artificial
Trabajo Practico 3

Nombre: Eduardo Echeverria (a1516)

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?

Para poder realizar la clasificación de forma mas sencilla, partimos de dataset agrupado por los valores de la columna “spam”, estamos hablando de la variable “column_sum”. Sobre este dataset, se realiza la transposición del mismo de forma que queden como columnas los valores “0” y “1” de “spam” y queden como filas los indicadores de cada palabra.

Luego se removieron las columnas de los indices y se obtuvieron los 10 valores mas altos de ambas columnas, es decir de los conjuntos de “spam” y “no spam”.

Todo esto se implemento en la siguiente sección de código:

```
# Removemos columnas de caracteres especiales porque no contribuyen al analisis
column_sum_no_char = column_sum.drop(['char_freq_;','char_freq_(','char_freq_[','char_freq_!','char_freq_$','char_freq_#'], axis=1)

# Se obtienen los 10 valores mas altos para el conjunto de "spam"
column_sum_spam = column_sum_no_char.transpose().drop(0, axis='columns')
top_spam_words = column_sum_spam.nlargest(10, 1)
top_spam_words.columns=['Word Occurence (SPAM)']
print(top_spam_words)

# Se obtienen los 10 valores mas altos para el conjunto de "no spam"
column_sum_no_spam = column_sum_no_char.transpose().drop(1, axis='columns')
top_nonspam_words = column_sum_no_spam.nlargest(10, 0)
top_nonspam_words.columns=['Word Occurence (Non-SPAM)']
print(top_nonspam_words)
```

Se obtuvieron los valores siguientes para cada conjunto:

Correos “Spam”

Palabra	Cantidad de veces que es utilizada
word_freq_you	4105599
word_freq_your	2502597
word_freq_will	997100
word_freq_free	939790
word_freq_our	931799
word_freq_all	732080
word_freq_mail	635470
word_freq_email	578759
word_freq_business	521250
word_freq_remove	499309

Correos “No - Spam”

Palabra	Cantidad de veces que es utilizada
word_freq_you	3541702
word_freq_george	3527559
word_freq_hp	2496576
word_freq_will	1495268
word_freq_your	1223098
word_freq_hpl	1204398
word_freq_re	1159138
word_freq_edu	800669
word_freq_address	681569
word_freq_meeting	604460

Se observa que algunas palabras son comunes en la lista de correos spam como en la de correos no-spam, por ejemplo “you”, “your”, “will”. Esto tiene sentido considerando que en inglés son palabras frecuentemente utilizadas para formas las oraciones mas comunes.

Quizás una de las palabras que llaman la atención, más que nada en el contexto de correos spam es la palabra “free”, la cuál es frecuentemente utilizada en correos de promociones y similares, los que casi siempre son correos no deseados.

En la lista correspondiente a correos legítimos, se encuentran palabras con un contexto un poco mas puntual o específico, por ejemplo “hp”, “hpl”, “edu”, “address”, “meeting” a diferencia de las palabras en la lista de correos spam que contienen palabras de interpretación general.

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

La separación del conjunto de datos de forma que un 70% de los datos sean el conjunto de entrenamiento y el 30% restante sean el conjunto de prueba se realiza mediante el siguiente código en Python:

```
# codigo del notebook ayuda
# Se prepara el dataset en entrenamiento y evaluacion

X_train, X_test, y_train, y_test= train_test_split(X, y, train_size = 0.7, test_size = 0.3)
```

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

Para este ejercicio, se utilizara un clasificador Naive-Bayes multinomial. Para entrenar con este modelo nuestro conjunto de entrenamiento realizamos lo siguiente en el código de Python:


```
# codigo del notebook ayuda
# Se prepara el dataset en entrenamiento y evaluacion

X_train, X_test, y_train, y_test= train_test_split(X, y, train_size = 0.7, test_size = 0.3)
```

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

Antes de aplicar el clasificador de Regresión Logística, se realiza primero el escalamiento de los conjuntos de entrenamiento y de prueba. Luego se los transforma a Data Frames y finalmente se aplica el clasificador.

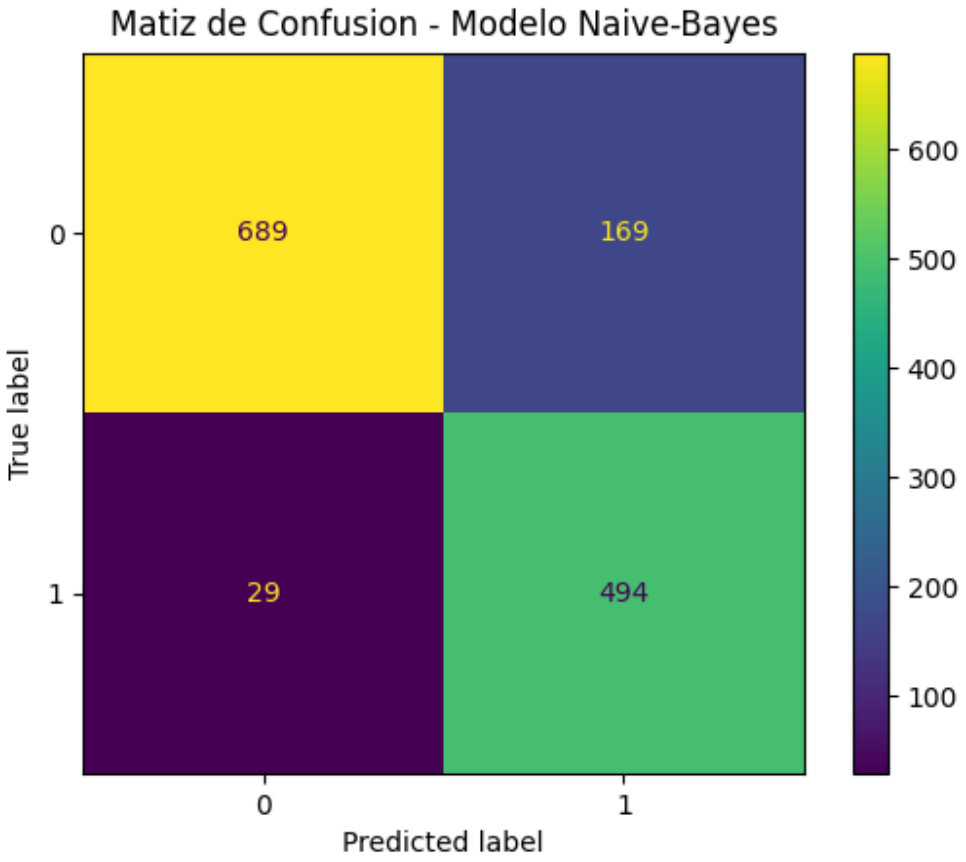
Todo este proceso viene implementado en la siguiente sección en el código de Python:

```
# codigo del notebook ayuda
# Escalamos para aplicar regresion logistica
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Lo transformamos en DataFrames
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X.columns)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X.columns)

regresion = LogisticRegression()
y_pred_reg = regresion.fit(X_train_scaled, y_train).predict(X_test)
```

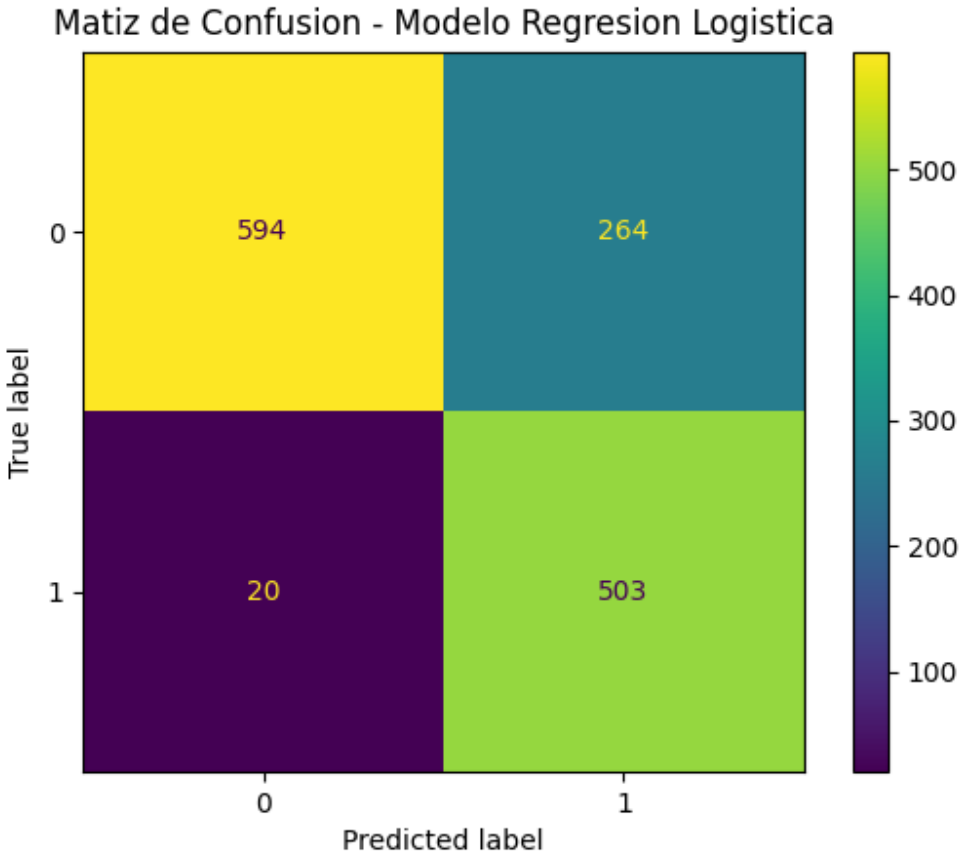
5) Calcule la matriz de confusión del conjunto de evaluación para ambos modelos. ¿Qué tipo de error comete más cada modelo? ¿Cuál de los dos tipos de error crees que es más importante para este problema?



Sobre el total del conjunto de evaluación, el modelo de Naive-Bayes predijo que 718 correos son en efecto correos legítimos, mientras que 663 son correos Spam. Sin embargo la clasificación “verdadera” indica que 858 correos son legítimos y 523 son Spam.

Con este modelo de clasificación, se obtuvieron los siguientes resultados:

- Verdaderos positivos (Spam, identificados como Spam): 494
- Falsos negativos (Spam, identificados como No-Spam): 29
- Falsos positivos (No-Spam, identificados como Spam): 169
- Verdaderos negativos (No-Spam, identificados como No-Spam): 689



El modelo de Regresión Logística identificó que del total del conjunto de evaluación, existen 614 correos legítimos y 767 correos Spam. Mientras que la clasificación "verdadera" identifica 858 correos legítimos y 523 correos Spam.

Con este modelo de clasificación, los resultados son los siguientes:

- Verdaderos positivos (Spam, identificados como Spam): 503
- Falsos negativos (Spam, identificados como No-Spam): 20
- Falsos positivos (No-Spam, identificados como Spam): 264
- Verdaderos negativos (No-Spam, identificados como No-Spam): 594

Conclusión: Se puede observar que el modelo de Regresión Logística tiene un mayor número de falsos positivos, es decir correos identificados como Spam pero que son en realidad correos legítimos. Mientras que el modelo de Naive-Bayes tiene una mayor cantidad de falsos negativos, es decir correos identificados como legítimos pero que en realidad son Spam.

El objetivo de la clasificación de los correos es precisamente identificar correos Spam para poder de esta forma evitarlos. Por tanto los errores que de forma equivocada identifican correos Spam como legítimos, es decir los falsos negativos son los errores mas importantes para este problema. Lo que nos indica que el clasificador de Naive-Bayes es el menos eficiente.

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 anterior? Expanda su respuesta.

Los resultados obtenidos para ambos modelos son los siguientes:

Precisión

De acuerdo a la definición de Precision contenida en la documentación de la librería scikit-learn, la precision indica la habilidad del clasificador de no etiquetar como positiva una muestra que es negativa, es decir mide la habilidad de evitar los falsos positivos[1].

Aplicando los modelos de Naive-Bayes y Regresión Logística obtenemos los siguientes valores de precisión:

- Precisión modelo Naive-Bayes es: 0.8676553897542635
- Precisión modelo de Regresión Logística es: 0.8283167279559904

Se observa que la precision del modelo Naive-Bayes es la mejor de los dos, lo cual concuerda con los resultados obtenidos en el punto anterior. Como se observa, el modelo de Naive-Bayes reportó 169 falsos positivos mientras que el modelo de Regresión Logística encontró 264 resultados que se determinaron como falsos positivos. En este sentido el Modelo de Naive-Bayes es mas eficiente para evitar los falsos positivos.

[1] Documentación Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

Recuperación

De acuerdo a la definición de recuperación encontrada en la documentación de Scikit-Library, la recuperación se puede definir como la habilidad del clasificador de encontrar todas las muestras positivas[2].

Para los modelos de Naiv-Bayes y Regresión Logística obtenemos los siguientes valores de recuperación:

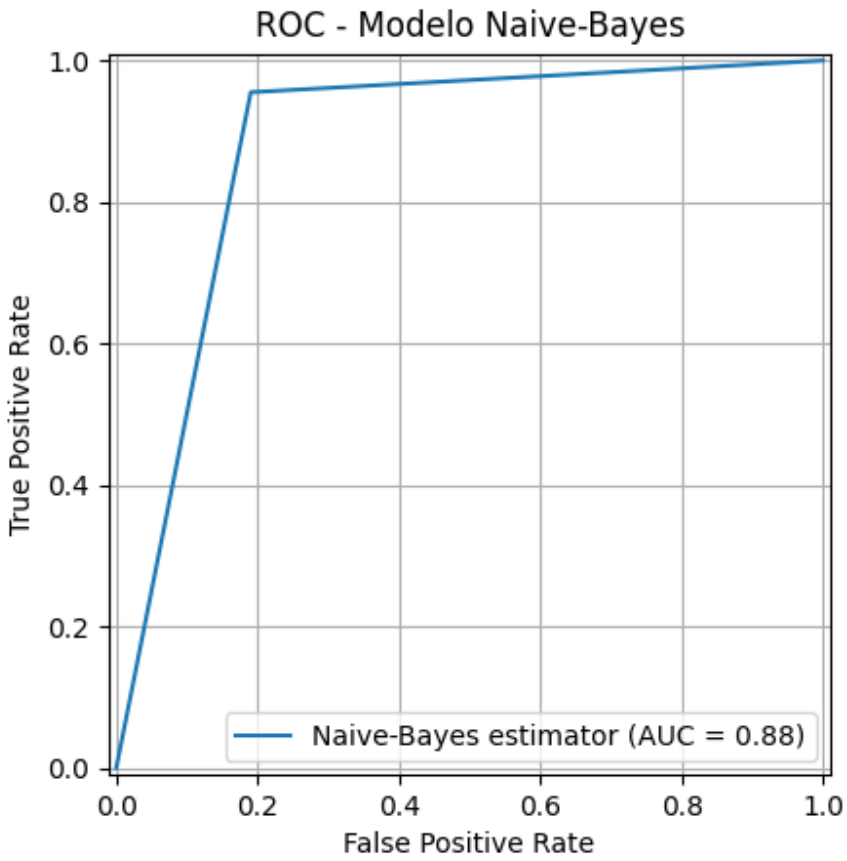
- Recuperación modelo Naive-Bayes es: 0.8861014282629485
- Recuperación modelo de Regresión Logística es: 0.8411184167037578

En este caso se observa que el modelo de Naive-Bayes muestra un mejor valor de recuperación comparado con el de Regresión Logística. Sin embargo comparando con el punto anterior vemos que ambos modelos obtienen un valor aproximado de verdaderos positivos.

[2] Documentación Scikit-Learn: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

7) Obtenga la curva ROC y el AUC (Área Bajo la Curva ROC) de ambos modelos.

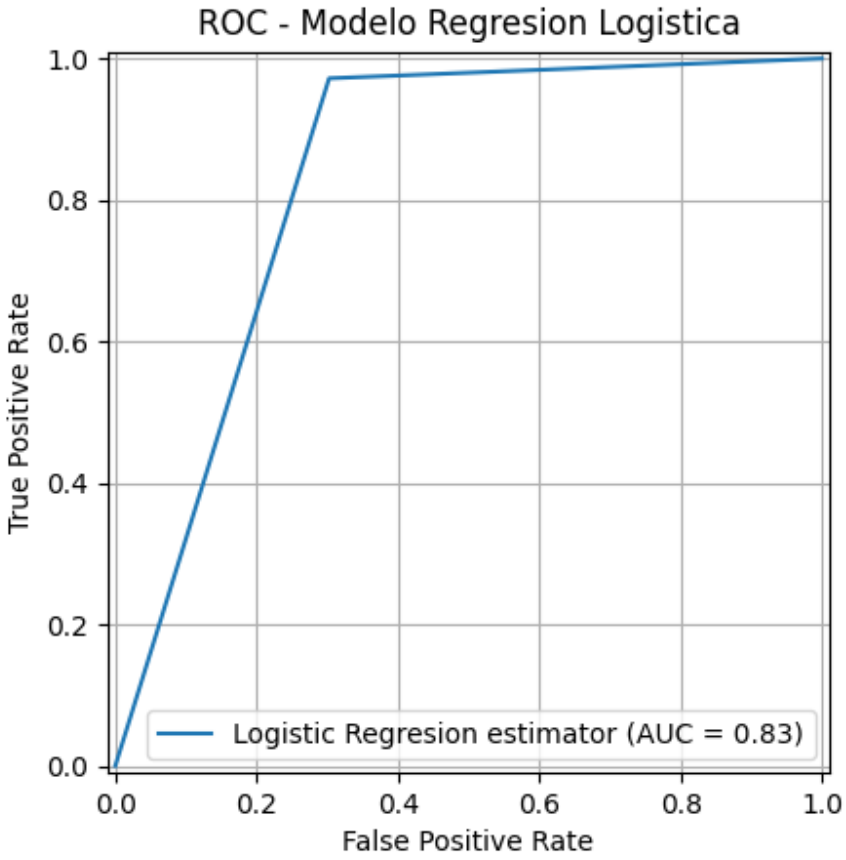
Para el modelo de Naive-Bayes tenemos lo siguiente:



Con la siguiente área bajo la curva:

- AUC para el modelo Naive-Bayes es: 0.8824164291553435

Para el modelo de Regresión Logística tenemos lo siguiente:



Con la siguiente área bajo la curva:

- AUC para el modelo de Regresión Logística es: 0.8346810720045956

Nota:
El repositorio de GitHub donde se encuentra el código es el siguiente:

https://github.com/eduardo-echeverria/entregas-ai/tree/main/clasificador_spam