**Taller 1**

Luis Santiago Jaramillo Espinosa, Parra Martínez Julio Alberto

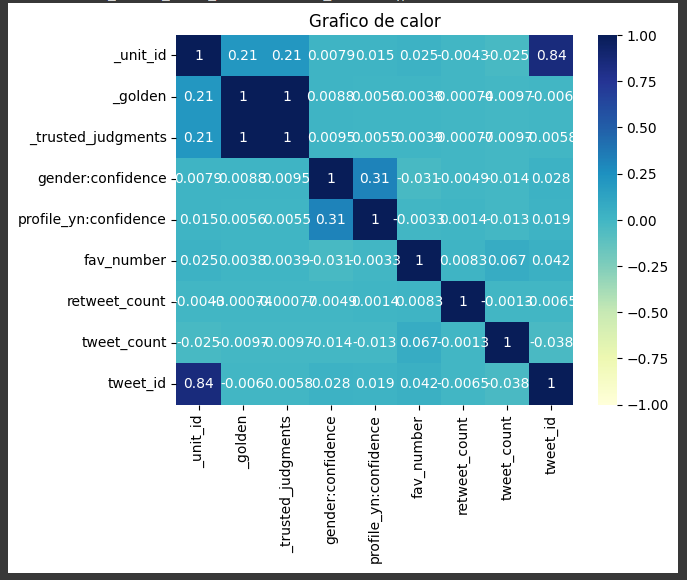
Facultad de ingeniería, Pontificia Universidad Javeriana de Bogotá

# Comprensión del dataset

**1.1 ¿Qué información presenta el dataset?**

El dataset consta de 16 columnas y 20.050 datos. Algunas de las variables son categóricas por lo que más adelante serán tratadas. Así también se también se tienen datos de gran aporte como lo son **description** que pueden ser tratadas con NLP para de este modo poder ser mas precisos en la predicción de **gender**.

Se presenta el grafico de correlaciones.



# Limpieza de los datos

Dado que el dataset tiene alrededor de aproximadamente 14.000 campos vacíos, no podemos eliminar todos aquellos campos ya que afectaría la precisión del modelo. Se opta por borrar algunos campos vacíos de algunas columnas, así como de eliminar columnas como lo son: '\_unit\_id','tweet\_id', 'user\_timezone', 'tweet\_coord', 'description', 'profileimage', 'text','link\_color', 'tweet\_location', 'name', '\_last\_judgment\_at', 'created','tweet\_created'. Así también se revisa el campo gender y se identifica que tiene 4 posibles valores: male, female, Brand y unknown. Se eliminan los Brand y los Unknowns ya que no aportan valor a la predicción de genero.

Finalmente de 26 columnas pasamos a tener solo 16: ['\_unit\_state', '\_trusted\_judgments', 'gender', 'gender:confidence', 'profile\_yn', 'profile\_yn:confidence', 'fav\_number', 'gender\_gold', 'profile\_yn\_gold', 'retweet\_count', 'sidebar\_color', 'tweet\_count']

# Variables Categoricas

Posteriormente se transforman male y female, donde male equivale a 1 y female a 0. Así también en las columnas 'profile\_yn' y 'profile\_yn\_gold' se reemplaza Yes por 1. A la columna de color 'sidebar\_color' se le asignan también valores numéricos.

# Construcción del dataset

**4.1 ¿Qué diferencia hay en usar un conjunto de validación?**

Muchos modelos son propensos a un problema llamado sobreajuste que es cuando el modelo se ajusta tan bien a los datos de entrenamiento que no se generaliza bien a los nuevos datos. El signo revelador del sobreajuste es un modelo que funciona extremadamente bien con los datos de entrenamiento y su rendimiento es inferior con nuevos datos. La forma de tener esto en cuenta es dividir el conjunto de datos en varios conjuntos: un conjunto de entrenamiento para entrenar el modelo, un conjunto de validación para comparar el rendimiento de diferentes modelos y un conjunto de prueba final para comprobar cómo funcionará el modelo en el mundo real. [1]

**4.2 ¿Mejora los resultados en la construcción del modelo usar un conjunto de validación?**

No mejora los resultados, pero nos permite hacer ajustes al modelo como la tasa de aprendizaje o el número de iteraciones de esta forma optimizando su rendimiento. [2]

**4.3 ¿Qué información provee el uso de un conjunto de validación?**

El conjunto de validación afecta a un modelo, pero sólo indirectamente por lo que no provee información.

# Elaboración del Modelo

**5.1 Perceptrón.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **from** **sklearn.linear\_model** **import** Perceptron  **from** **sklearn.metrics** **import** confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score  **from** **sklearn.metrics** **import** confusion\_matrix, ConfusionMatrixDisplay  **import** **matplotlib.pyplot** **as** **plt**  #Distribucion de data recalcular,  #cambiar manejod e data comparaciion  # Entrenar el perceptrón  perceptron = Perceptron(max\_iter=**1000**, eta0=**0.01**)  perceptron.fit(X, y)  # Predecir las etiquetas  y\_pred\_1 = perceptron.predict(X)  # Calcular la matriz de confusión  matriz\_confusion\_1 = confusion\_matrix(y, y\_pred\_1)  # Calcular las métricas de evaluación  precision\_1 = precision\_score(y, y\_pred\_1)  recall\_1 = recall\_score(y, y\_pred\_1)  f1\_1 = f1\_score(y, y\_pred\_1) |

**5.2 Red Neuronal con una capa oculta con un numero de neurona igual al número de entradas.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **from** **sklearn.neural\_network** **import** MLPClassifier  # Entrenar la red neuronal, activation default='relu'  mlp = MLPClassifier(hidden\_layer\_sizes=(X.shape[**1**],), max\_iter=**1000**, solver='adam')  mlp.fit(X, y)  # Predecir las etiquetas  y\_pred\_2 = mlp.predict(X)  # Calcular la matriz de confusión  matriz\_confusion\_2 = confusion\_matrix(y, y\_pred\_2)  # Calcular las métricas de evaluación  precision\_2 = precision\_score(y, y\_pred\_2)  recall\_2 = recall\_score(y, y\_pred\_2)  f1\_2 = f1\_score(y, y\_pred\_2) |

**5.3 Red Neuronal con dos capas ocultas, la primera con las mitades de las entradas y la segunda con la misma cantidad de la capa oculta, la tercera capa será la de salida.**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

**from** **sklearn.preprocessing** **import** LabelEncoder

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **tensorflow.keras.models** **import** Sequential

**from** **tensorflow.keras.layers** **import** Dense

# Dividir los datos en entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test\_3 = train\_test\_split(X, y,train\_size=**0.6**,test\_size=**0.4**, random\_state=**42**)

# Definir la arquitectura de la red neuronal

model = Sequential()

# Primera capa oculta

model.add(Dense(X\_train.shape[**1**] // **2**, activation='relu', input\_shape=(X\_train.shape[**1**],)))

# Segunda capa oculta

model.add(Dense(X\_train.shape[**1**] // **2**, activation='relu'))

# Capa de salida

model.add(Dense(**1**, activation='sigmoid'))

# Compilar el modelo

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Entrenar el modelo

model.fit(X\_train, y\_train, epochs=**100**)

# Predecir las etiquetas

y\_pred\_3 = model.predict(X\_test)

# Calcular la matriz de confusión

matriz\_confusion\_3 = confusion\_matrix(y\_test\_3, y\_pred\_3)

# Calcular las métricas de evaluación

precision\_3 = precision\_score(y\_test\_3, y\_pred\_3)

recall\_3 = recall\_score(y\_test\_3, y\_pred\_3)

f1\_3 = f1\_score(y\_test\_3, y\_pred\_3)

# Análisis de Resultados

# Ajustes

# Referencias

[1] “Cómo preparar un conjunto de datos para machine learning y análisis”, en *datos.gob.es, 2023*. Consultado: el 18 de febrero de 2023. [En línea]. Disponible en: <https://datos.gob.es/es/blog/como-preparar-un-conjunto-de-datos-para-machine-learning-y-analisis>

[2] Shah Tarang, “About Train, Validation and Test Sets in Machine Learning”, en Medium 2017. Consultado: el 18 de febrero de 2023. [En línea]. Disponible en: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>