

Detecting YouTube Comment Spam

Equipo

Emiliano Lara Huerta/A00573433

Karen Gutiérrez Aguilar/A00572511

Galilea Oaxaca Moreno/A00572347

- **Lectura de caso:** Detecting YouTube comment spam (Pág.51).
Libro: Python Artificial Intelligence Projects for Beginners

- **Introducción:** A partir de varios DataSets con comentarios de YouTube, los cuales nos dicen que comentarios contienen spam, y cuáles no, vamos a utilizar varias técnicas de Ciencia de Datos para crear un modelo que nos permita saber con comentarios que no estén clasificados, cuales son Spam y Cuáles no: principalmente usaremos Count Vectorizer, que ayuda a contar cuántas palabras aparecen dentro de un vector; bolsas de palabras, árboles de decisión y random forest, entre otras cosas.

- **Glosario semana 1:**
 - **DataSet:** un Dataset o conjunto de datos hace relación (en su versión más simple) a los contenidos de una única tabla de bases de datos donde cada columna de la tabla representa una variable en particular y cada fila representa a un miembro determinado del conjunto de datos.

 - **Scikit-learn:** Scikit-learn es una biblioteca de aprendizaje automático de software libre para el lenguaje de programación Python.

 - **Bolsa de palabras:** o Count Vectorizer, es la herramienta que nos permite separar los comentarios en palabras individuales, convirtiéndolas en vectores.

Problemas que tuvimos:

- El link para abrir la Dataset no funciona bien, es necesario encontrar una diferente a la que sugiere el libro, con la misma temática.

Definición de roles:

Bitácora- Emiliano Lara Huerta

Investigación- Karen Gutiérrez Aguilar

Trabajo en Python- Galilea Oaxaca Moreno

Definición de horario de trabajo:

- **Martes, miércoles y viernes** de 11:00 a.m. a 1:00 p.m.
- **Lunes y jueves** de 4:00 a 5:00 p.m.

Para la próxima fecha:

Generaremos la base de datos, verificaremos si existe la librería y en caso de que no sea así, la cambiaremos por otra.

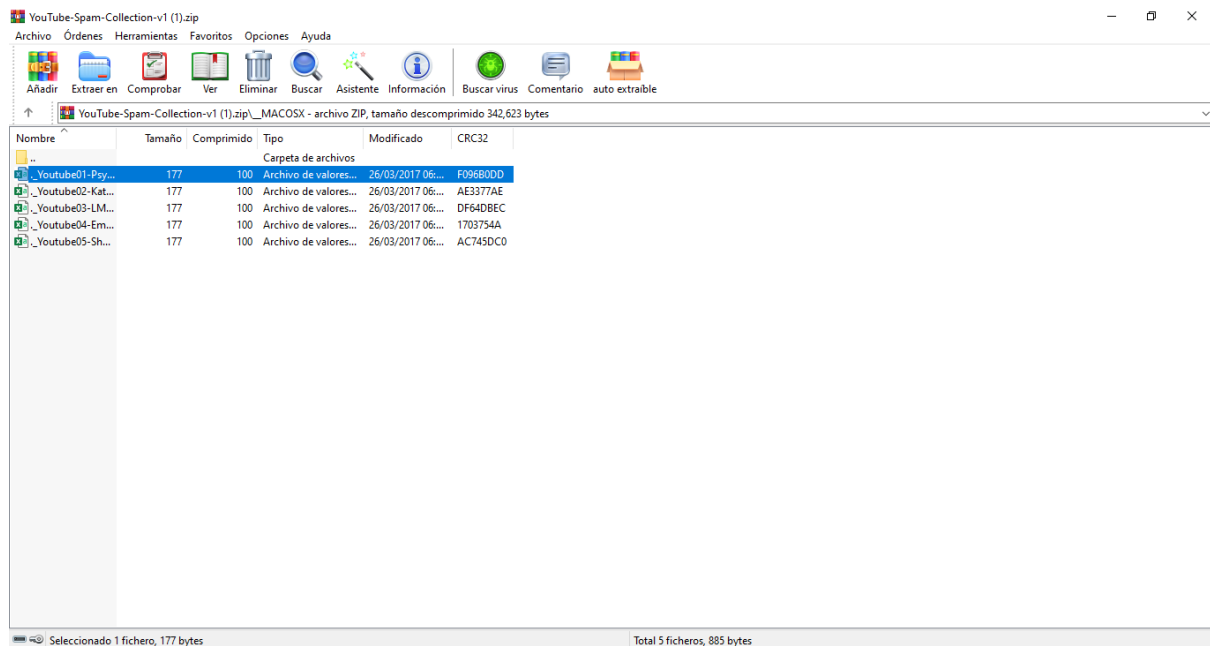
Avance 19 de mayo del 2022

[Presentación 19/05/22](#)

Básicamente, para este día resolvimos el problema de los DataSets, y por fin los pudimos descargar.

Avance 23 de mayo del 2022

Para este avance buscamos verificar el funcionamiento de la biblioteca y la base de datos. Pudimos abrir el DataSet, solo tendríamos que revisar los datos para ver cuales son funcionales y cuáles no.



También investigamos algunas librerías que pudiéramos necesitar, para ver si podríamos utilizarlas.

Para la siguiente clase:

- Analizar la lectura del problema para revisar todos los conceptos y procedimientos que no comprendamos.
- Analizar la base de datos, para ver qué elementos sirven y cuales pueden ser omitidos.
- Preparar el entorno y las librerías.

Glosario (análisis de conceptos y procedimientos no comprendidos):

Bags of words: es un modelo en el procesamiento del lenguaje, que separa las palabras, ignorando su orden en una frase, por lo que parece una “bolsa” que contiene algunas palabras. CountVectorizer es el nombre de esta técnica en scikit-learn.

Vector: un vector, array o arreglo (un identificador que referencia a un grupo de elementos del mismo tipo), es lo que se conoce comúnmente en Python como una lista. Es una estructura de datos utilizada para almacenar múltiples valores en una única variable.

Random forest: Es un algoritmo que analiza posibles resultados de una serie de datos relacionados entre sí. Se comienza con un par de datos, que se van ramificando, para generar clasificaciones, o para la toma de decisiones dependiendo del curso que tomen las ramificaciones del árbol. En Machine Learning, se generan múltiples árboles de decisión, generando de una base de datos un “random forest”, que permite tener múltiples opciones de decisiones o clasificaciones para una base de datos.

Split the dataset into training and testing sets: es una técnica que consiste en separar el “dataset”, para después, con algunas secciones “entrenarla” en alguna función o proceso y “probarlo” con las otras partes del DataSet. La base de datos se divide en varias partes, para evitar usar los mismos datos para testear y para entrenar.

Cross validation: La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Como tenemos 5 DataSets de diferentes videos, comprobaremos con un algoritmo de “Cross validation”, el buen funcionamiento del random forest generado del primer “training and test”. Usaremos DataSet para entrenar, y uno para testearlos.

Pipeline: en Machine Learning, es la construcción que organiza el flujo de datos, hacia un modelo de aprendizaje automático (o un conjunto de múltiples modelos) y la salida de este. Incluye entrada de datos sin procesar, características, salidas, el modelo de aprendizaje automático y parámetros del modelo, y salidas de predicción. Mejora la confianza en la circulación de datos de un modelo.

TF-IDF: Está es una medida numérica que expresa qué tan relevante es una palabra para un documento en una colección. Esta medida se utiliza comúnmente como un factor de ponderación en la recuperación de información.

ngrams: Un n-grama es una sub secuencia de n elementos de una secuencia dada. Este es usado para distintas áreas del conocimiento como en el estudio del lenguaje natural, en el estudio de las secuencias de genes, etc.

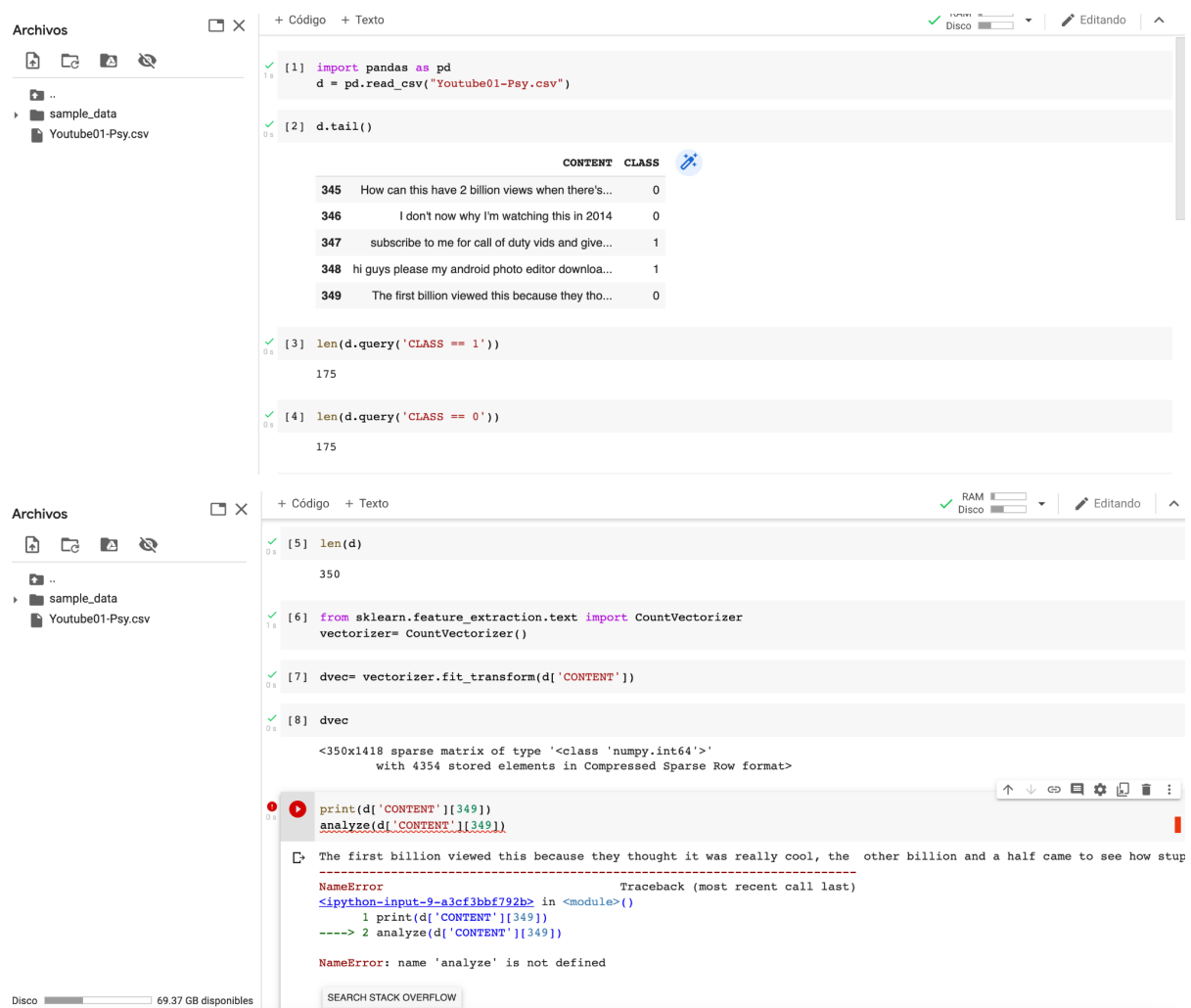
En el estudio del lenguaje natural podríamos construir los n-gramas sobre la base de distintos tipos de elementos como: fonemas, sílabas, palabras, letras, entre otros.

Parameter search feature known as grid search.

Grid searching es un método utilizado en Python para encontrar la mejor combinación posible de hiper parámetros (valores ¿?) en los que el modelo logre la máxima precisión.

Trabajo en la base de datos

Tenemos **5 datasets** con comentarios de diferentes videos; para la primera fase del problema solo ocuparemos una de las cinco, por lo que limpiamos las columnas que no ocupamos (COMMENT ID y DATE) para dejar solo el contenido del comentario y si es spam o no. Ya pudimos **integrar la base de datos en el entorno**, para continuar con los siguientes pasos.



The screenshot shows a Jupyter Notebook interface with two visible cells. The first cell contains code to load a CSV file and display its tail. The second cell contains code to filter the data by class and calculate the length of each subset. A third cell shows the creation of a CountVectorizer and its application to the 'CONTENT' column. A fourth cell shows an attempt to use a function named 'analyze' which results in a NameError.

```
[1] import pandas as pd
d = pd.read_csv("Youtube01-Psy.csv")

[2] d.tail()
```

	CONTENT	CLASS
345	How can this have 2 billion views when there's...	0
346	I don't now why I'm watching this in 2014	0
347	subscribe to me for call of duty vids and give...	1
348	hi guys please my android photo editor downloa...	1
349	The first billion viewed this because they tho...	0

```
[3] len(d.query('CLASS == 1'))
175

[4] len(d.query('CLASS == 0'))
175

[5] len(d)
350

[6] from sklearn.feature_extraction.text import CountVectorizer
vectorizer= CountVectorizer()

[7] dvec= vectorizer.fit_transform(d['CONTENT'])

[8] dvec
<350x1418 sparse matrix of type '<class 'numpy.int64'>'
with 4354 stored elements in Compressed Sparse Row format>

print(d['CONTENT'][349])
analyze(d['CONTENT'][349])

NameError                                Traceback (most recent call last)
<ipython-input-9-a3cf3bbf792b> in <module>()
      1 print(d['CONTENT'][349])
----> 2 analyze(d['CONTENT'][349])

NameError: name 'analyze' is not defined
```

Usamos la plataforma *Google Collab* para poder trabajar en equipo. Nuestro proceso fue el siguiente:

- 1) Importar la librería
- 2) Imprimir los comentarios y limpiar la información

- 3) Sacar el número de comentarios spam y el número de comentarios que no son spam (1 → spam, 0 → no spam)
- 4) Número total de comentarios
- 5) Contar cuantas veces aparece cada palabra y ponerlas en un vector
- 6) 350 filas= 350 comentarios. 1418 palabras
- 7) Tratamos de dividir una frase en cada palabra pero nos sale que la palabra “analyze” no está definida

Avance 30 de mayo del 2022

- **Introducción** de la base de datos a Python.
- **Contar** cuántos de los comentarios son spam y cuántos no.
- **Count Vectorizer** para contar las palabras de los comentarios, y ponerlas en vectores, que usaremos para entrenar nuestro random forest (algoritmo que analiza posibles resultados de una serie de datos relacionados entre sí).
- **Separar el DataSet** en set de “training” (300 comentarios) y de “test” (50 comentarios). Usamos las palabras de set de entrenamiento para el random forest.
- **Convertimos el DataSet en 80 árboles** que posteriormente al entrenamiento, podremos “probarlos” para revisar su desempeño.


```

In [48]: import pandas as pd
In [49]: d = pd.read_csv("Youtube01-Psy.csv")
In [50]: d.tail()
Out[50]:
          CONTENT  CLASS
345  How can this have 2 billion views when there's...      0
346  I don't now why I'm watching this in 2014      0
347  subscribe to me for call of duty vids and give...      1
348  hi guys please my android photo editor downloa...      1
349  The first billion viewed this because they tho...      0

In [51]: len(d.query('CLASS == 1'))
Out[51]: 175

In [52]: len(d.query('CLASS == 0'))
Out[52]: 175

In [53]: len(d)
Out[53]: 350

In [54]: from sklearn.feature_extraction.text import CountVectorizer
In [55]: vectorizer= CountVectorizer()
In [56]: dvec= vectorizer.fit_transform(d['CONTENT'])
In [57]: dvec
Out[57]:
<350x1418 sparse matrix of type '<class 'numpy.int64'>'
      with 4354 stored elements in Compressed Sparse Row format>

In [58]: print(d['CONTENT'][349])
The first billion viewed this because they thought it was really cool, the other billion and a half came to see how stupid the first billion were...

In [59]: analyze(d['CONTENT'][349])

```

```

Out[59]:
['the',
 'first',
 'billion',
 'viewed',
 'this',
 'because',
 'they',
 'thought',
 'it',
 'was',
 'really',
 'cool',
 'the',
 'other',
 'billion',
 'and',
 'half',
 'came',
 'to',
 'see',
 'how',
 'stupid',
 'the',
 'first',
 'billion',
 'were']

In [60]: print(d['CONTENT'][349])
The first billion viewed this because they thought it was really cool, the other billion and a half came to see how stupid the first billion were...

In [61]: analyze=vectorizer.build_analyzer()

```

```

In [62]: analyze(d['CONTENT'])[349]
Out[62]:
['the',
 'first',
 'billion',
 'viewed',
 'this',
 'because',
 'they',
 'thought',
 'it',
 'was',
 'really',
 'cool',
 'the',
 'other',
 'billion',
 'and',
 'half',
 'came',
 'to',
 'see',
 'how',
 'stupid',
 'the',
 'first',
 'billion',
 'were']

```

```

In [64]: dshuf=d.sample(frac=1)

In [65]: d_train=dshuff[:300]
Traceback (most recent call last):

  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_40540/1947916062.py", line 1, in <module>
    d_train=dshuff[:300]
NameError: name 'dshuff' is not defined

In [66]: d_train=dshuf[:300]

In [67]: d_test=dshuf[300:]

In [68]: d_train_att=vectorizer.fit_transform(d_train['CONTENT'])

In [69]: d_test_att=vectorizer.transform(d_test['CONTENT'])

In [70]: cd_test_label=d_TEST['CLASS']
Traceback (most recent call last):

  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_40540/3359251530.py", line 1, in <module>
    cd_test_label=d_TEST['CLASS']
NameError: name 'd_TEST' is not defined

In [71]: d_test_label=d_test['CLASS']

In [72]: d_train_att
Out[72]:
<300x1276 sparse matrix of type '<class 'numpy.int64'>'
  with 3767 stored elements in Compressed Sparse Row format>

In [73]: d_test_att
Out[73]:
<50x1276 sparse matrix of type '<class 'numpy.int64'>'
  with 441 stored elements in Compressed Sparse Row format>

In [74]: from sklearn.ensemble import RandomForestClassifier

In [75]: clf= RandomForestClassifier(n_estimators=80)

In [76]: clf.fit(d_train_att, d_train_label)
Out[76]: RandomForestClassifier(n_estimators=80)

```

```

In [39]: clf.score(d_test_att, d_test_label)
Out[39]: 0.98

```

Pudimos observar además, que después de este procedimiento nos arrojó un nivel de confianza en el test de 98% (muy bueno).

- Vamos a crear un **Pipeline** para hacer un macro de pasos. El Pipeline incluye entrada de datos sin procesar, características, salidas, el modelo de aprendizaje automático, parámetros del modelo, y salidas de predicción. Mejora la confianza en la circulación de datos de un modelo.
- En el In [48] hay un error, que después sabríamos que ocurrió por no introducir los 5 Data Sets necesarios.

```
In [33]: from sklearn.metrics import confusion_matrix
In [34]: pred_labels=clf.predict(d_test_att)
In [35]: confusion_matrix(d_test_label, pred_labels)
Out[35]:
array([[29,  0],
       [ 0, 21]])

In [36]: from sklearn.model_selection import cross_val_score
In [37]: scores=cross_val_score(clf, d_train_att, d_train_label, cv=5)
In [38]: print ("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.96 (+/- 0.02)

In [39]: d= pd.concat([pd.read_csv("Youtube01-Psy.csv")])

In [40]: len(d)
Out[40]: 350

In [41]: len(d.query('CLASS == 1'))
Out[41]: 175

In [42]: len(d.query('CLASS== 0'))
Out[42]: 175

In [43]: dshuf = d.sample(frac=1)
In [44]: d_content=dshuf['CONTENT']
In [45]: d_label=dshuf['CLASS']
```

```

In [46]: from sklearn.pipeline import Pipeline, make_pipeline

In [47]: pipeline= Pipeline([
...: ('bag-of-words', CountVectorizer()),
...: ('random forest', RandomForestClassifier()),
...: ])

In [48]: pipeline
Out[48]:
Pipeline(steps=[('bag-of-words', CountVectorizer()),
                 ('random forest', RandomForestClassifier())])

In [49]: make_pipeline(CountVectorizer(), RandomForestClassifier())
Out[49]:
Pipeline(steps=[('countvectorizer', CountVectorizer()),
                 ('randomforestclassifier', RandomForestClassifier())])

In [50]: pipeline.fit(d_content[:1500],d_label[:1500])
Out[50]:
Pipeline(steps=[('bag-of-words', CountVectorizer()),
                 ('random forest', RandomForestClassifier())])

```

```

In [51]: pipeline.score(d_content[1500:], d_label[1500:])
Traceback (most recent call last):

  File "/var/folders/tp/3_g5b-w07b99dk2828rthcsfh08000gn/T/ipykernel_50589/2960104310.py", line 1, in <module>
    pipeline.score(d_content[1500:], d_label[1500:])

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/metaestimators.py", line 120, in
<lambda>
    out = lambda *args, **kwargs: self.fn(obj, *args, **kwargs)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/pipeline.py", line 622, in score
    return self.steps[-1][-1].score(Xt, y, **score_params)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py", line 500, in score
    return accuracy_score(y, self.predict(X), sample_weight=sample_weight)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py", line 630, in
predict
    proba = self.predict_proba(X)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py", line 676, in
predict_proba
    X = self._validate_X_predict(X)

```

```

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_forest.py", line 422, in
_validate_X_predict
    return self.estimators_[0]._validate_X_predict(X, check_input=True)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/tree/_classes.py", line 407, in
_validate_X_predict
    X = self._validate_data(X, dtype=DTYPE, accept_sparse="csr",

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py", line 421, in _validate_data
    X = check_array(X, **check_params)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py", line 63, in
inner_f
    return f(*args, **kwargs)

  File "/Users/gallioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/validation.py", line 726, in
check_array
    raise ValueError("Found array with %d sample(s) (shape=%s) while a"

ValueError: Found array with 0 sample(s) (shape=(0, 1418)) while a minimum of 1 is required.

In [52]:

```

Avance 6 de junio del 2022

En la presentación además de nuestro código, vemos algunos conceptos que no habían quedado del todo claros: Árboles de decisión, matrix confusión, 80-20 en estadística y validación cruzada:

[Presentación 6/06/22](#)

- Después de no incluir los otros DataSets en el código en el avance pasado, esta vez los concatenamos en un solo data frame.
- Para los siguientes pasos todo iba bien, pero al momento de crear el Pipeline con la bolsa de palabras y el random forest classifier, nos dio un resultado diferente al que nos aparece en el libro, por lo que pensamos que era un error. Después sabríamos que a pesar de que el resultado no era el predicho por el libro que consultamos, podíamos continuar con la base de datos.

Para la siguiente clase:

Seguiremos haciendo la base de datos, esperando terminar el código.

```
Console 1/A x
...: from sklearn.model_selection import cross_val_score
...: scores=cross_val_score(clf, d_train_att, d_train_label, cv=5)
...: print ("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
...: d= pd.concat([pd.read_csv("Youtube01-Psy.csv"), pd.read_csv("Youtube002-KatyPerry.csv"),
pd.read_csv("Youtube003-LMFAO.csv"), pd.read_csv("Youtube004-Eminem.csv"), pd.read_csv("Youtube005-
Shakira.csv")])
...: len(d)
The first billion viewed this because they thought it was really cool, the other billion and a half came to
see how stupid the first billion were...
Accuracy: 0.94 (+/- 0.05)
Out[35]: 1956

In [36]: len(d.query('CLASS == 1'))
Out[36]: 1005

In [37]: len(d.query('CLASS== 0'))
Out[37]: 951

In [38]: dshuf = d.sample(frac=1)

In [39]: d_content=dshuf['CONTENT']

In [40]: d_label=dshuf['CLASS']

In [41]: from sklearn.pipeline import Pipeline, make_pipeline
```

```

Out[37]: 951

In [38]: dshuf = d.sample(frac=1)

In [39]: d_content=dshuf['CONTENT']

In [40]: d_label=dshuf['CLASS']

In [41]: from sklearn.pipeline import Pipeline, make_pipeline

In [42]: pipeline= Pipeline([('bag-of-words', CountVectorizer()),('random forest',
RandomForestClassifier()),])

In [43]: pipeline
Out[43]:
Pipeline(steps=[('bag-of-words', CountVectorizer()),
                 ('random forest', RandomForestClassifier())])

In [44]: pipeline()
Traceback (most recent call last):

  File "C:\Users\emila\AppData\Local\Temp\ipykernel_43432\1150355636.py", line 1, in <module>
    pipeline()
TypeError: 'Pipeline' object is not callable

```

Avance 9 de junio del 2022

- Clasificación de comentarios como spam o no spam, 2 ejemplos.
- Comprobar la precisión de las respuestas.
- Finalmente terminamos el código; no logramos concluir hasta los últimos pasos, que servían para determinar la confiabilidad del programa, pero al final servía para realizar su propósito..

```

The first billion viewed this because they thought it was really cool, the other billion and a half came to see how stupid the first
billion were...
Accuracy: 0.96 (+/- 0.05)
Out[2]: 0.956140350877193

In [3]: pipeline.predict(["what a neat video!"])
Out[3]: array([0])

In [4]: pipeline.predict(["plz subscribe to my channel"])
Out[4]: array([1])

In [5]: scores = corss_val_score(pipeline, d_content, d_label, cv=5)
Traceback (most recent call last):

  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/1916930375.py", line 1, in <module>
    scores = corss_val_score(pipeline, d_content, d_label, cv=5)
NameError: name 'corss_val_score' is not defined

In [6]: scores = cross_val_score(pipeline, d_content, d_label, cv=5)

In [7]: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
Accuracy: 0.96 (+/- 0.01)

In [8]: from sklearn.feature_extraction.text import TfidfTransformer
Traceback (most recent call last):

  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/521922831.py", line 1, in <module>
    from sklearn.feature_extraction.text import TfidfTransformer
ImportError: cannot import name 'TfidfTransformer' from 'sklearn.feature_extraction.text' (/Users/galioaxaca/opt/anaconda3/lib/python3.9
site-packages/sklearn/feature_extraction/text.py)

```

```

In [9]: from sklearn.feature_extraction.text import TfidfTransformer
Traceback (most recent call last):

  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/521922831.py", line 1, in <module>
    from sklearn.feature_extraction.text import TfidfTransformer

ImportError: cannot import name 'TfidfTransformer' from 'sklearn.feature_extraction.text' (/Users/galioaxaca/opt/anaconda3/lib/python3.
site-packages/sklearn/feature_extraction/text.py)

In [10]: from sklearn.feature_extraction.text import TfidfTransformer pipeline2= make_pipeline(CpntVectorizer(),
TfidfTransformer(norm=None), RandomForestClassifier())
  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/3921724685.py", line 1
    from sklearn.feature_extraction.text import TfidfTransformer pipeline2= make_pipeline(CpntVectorizer(), TfidfTransformer(norm=None
RandomForestClassifier())
                                         ^
SyntaxError: invalid syntax

In [11]: from sklearn.feature_extraction.text import TfidfTransformer pipeline2= make_pipeline(CountVectorizer(),
TfidfTransformer(norm=None), RandomForestClassifier())
  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/8096393.py", line 1
    from sklearn.feature_extraction.text import TfidfTransformer pipeline2= make_pipeline(CountVectorizer(), TfidfTransformer(norm=None
RandomForestClassifier())
                                         ^
SyntaxError: invalid syntax

```

```

In [20]: from sklearn.model_selection import GridSearchCV

In [21]: grid_search=GridSearchCV(pipeline2, parameters, n_jobs=-1, verbose=1)

In [22]: grid_search.fit(d_content, d_label)
Fitting 5 folds for each of 72 candidates, totalling 360 fits
_RemoteTraceback:
"""
Traceback (most recent call last):
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/joblib/externals/loky/process_executor.py", line 436, in
_process_worker
    r = call_item()
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/joblib/externals/loky/process_executor.py", line 288, in __call__
    return self.fn(*self.args, **self.kwargs)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/joblib/_parallel_backends.py", line 595, in __call__
    return self.func(*args, **kwargs)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/joblib/parallel.py", line 262, in __call__
    return [func(*args, **kwargs)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/joblib/parallel.py", line 262, in <listcomp>
    return [func(*args, **kwargs)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/fixes.py", line 222, in __call__
    return self.function(*args, **kwargs)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_validation.py", line 586, in _fit_and_sc
    estimator = estimator.set_params(**cloned_parameters)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/pipeline.py", line 150, in set_params
    self._set_params('steps', **kwargs)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/metaestimators.py", line 54, in _set_params
    super().set_params(**params)
  File "/Users/galioaxaca/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py", line 230, in set_params
    raise ValueError('Invalid parameter %s for estimator %s.'
ValueError: Invalid parameter countvectorizer_max_features for estimator Pipeline(steps=[('countvectorizer', CountVectorizer()),
('tfidftransformer', TfidfTransformer(norm=None)),
('randomforestclassifier', RandomForestClassifier())]). Check the list of available parameters with
`estimator.get_params().keys()`.
"""

```

```

ValueError: Invalid parameter countvectorizer_max_features for estimator Pipeline(steps=[('countvectorizer', CountVectorizer()),
('tfidftransformer', TfidfTransformer(norm=None)),
('randomforestclassifier', RandomForestClassifier())]). Check the list of available parameters with
`estimator.get_params().keys()`.

In [23]: print("Best score: %0.3f" % grid_search.best_score_)
Traceback (most recent call last):

  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/3604662991.py", line 1, in <module>
    print("Best score: %0.3f" % grid_search.best_score_)

AttributeError: 'GridSearchCV' object has no attribute 'best_score_'

In [24]: print("Best score: %0.3f" % grid_search.best_score_) print ("Best parameters set:")
best_parameters=grid_search.best_estimator_.get_params() for param_name in sorted(parameters.keys()): print("\t%s: %r" % (param_name,
best_parameters[param_name]))
  File "/var/folders/fp/3_q30w0j7b9dk2020rbhzsfh0000gn/T/ipykernel_71347/2451716492.py", line 1
    print("Best score: %0.3f" % grid_search.best_score_) print ("Best parameters set:")
best_parameters=grid_search.best_estimator_.get_params() for param_name in sorted(parameters.keys()): print("\t%s: %r" % (param_name,
best_parameters[param_name]))
                                         ^
SyntaxError: invalid syntax

In [25]:

```

Conclusiones personales:

Galilea: este proyecto me ayudó a entender mejor todos los códigos que habíamos visto en clase. Al verlos de manera práctica en un ejercicio aplicado en una plataforma con la que estoy familiarizada (Youtube en este caso), pude entender para qué servía cada paso. También investigar las definiciones de ciertos términos, me ayudó a complementar mi aprendizaje sobre este tema.

Mi conclusión es que aunque al principio pensaba que programación no me iba a servir para mi carrera, hoy me doy cuenta de que estaba mal. Programación puede servir en distintas áreas laborales.

Emiliano: me ayudó a poner en práctica las fases para realizar un proyecto de Ciencia de Datos en equipo; me gusto poder elegir entre varias opciones de proyectos, para desarrollar el que mejor fuera con mis intereses. Se nos presentaron muchos retos y problemas, que al final pudimos resolver trabajando de la mano con los otros equipo de mis compañeros y haciendo investigación propia, por lo que me di cuenta que un proyecto de datos, en general puede resolverse con ayuda de una comunidad, sin necesidad de ser completos expertos en la materia.

Con un proyecto así, además aprendimos un montón de conceptos, herramientas y trucos que fueron necesarios o facilitaron mucho nuestro trabajo. Me resultó muy útil para desarrollar proyectos en los que quieras clasificar cosas dependiendo de lo que está escrito sin necesidad de revisar y leer todo un documento o una base de datos.

Karen:

Este proyecto me gustó y me ayudó mucho a poder aplicar mis conocimientos adquiridos en clase en un problema de la vida real. Fue un proyecto muy completo, un proyecto en el que aprendí no solo códigos de programación en Python, sino también nueva terminología y conceptos sobre la ciencia de datos que me permitieron comprender completamente nuestro problema y que se que me serán muy útiles en un futuro.

En conclusión, salgo de esta clase contenta, sabiendo que adquiriré muchos nuevos conocimientos que me sumarán mucho tanto a mi vida laboral como a mi vida personal.