

Alejandro Aceves Gutiérrez A00573606
Ángel David Chacón Fajardo A00573363
Gabriel Venegas Aguilera A00572185
Mariano Dávila Martínez A00572493

15 de junio del 2022
Bitácora final

Detectar comentarios spam en youtube

Tecnológico de Monterrey

Profesor Jorge Dávila

Matemáticas y ciencia de datos para la toma de decisiones



**Tecnológico
de Monterrey**

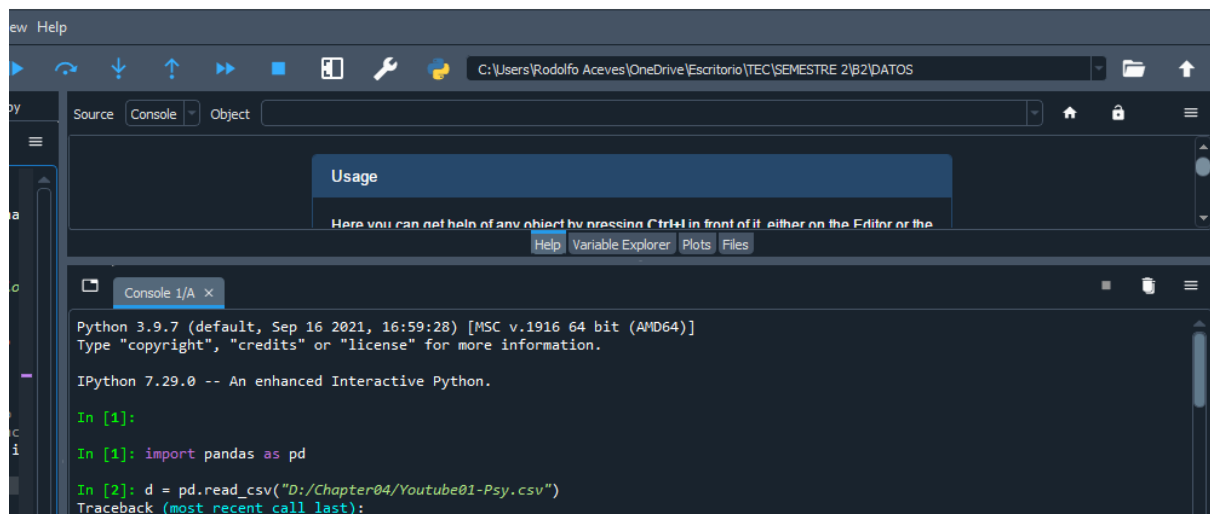
LIBRO

<https://drive.google.com/file/d/1i0Aq8s3vF8e8KBupsHEd69IDRAV0WQD9/view?usp=sharing>

Bitácora semana 1: Leímos el libro de Python for beginners, escogimos el problema sobre YouTube y los comentarios de spam, empezamos a hacer la presentación y nos repartimos qué trabajos y tareas iremos haciendo por cada semana, vamos bien, hemos entendido el problema y buenas partes del código, pero tenemos que resolver lo que no entendemos o lo que no salga del código, por eso necesitaremos ayuda de el profesor.

- Hacer tabla, 1 y 0
- Escoger vídeo pequeño
- tenerlo en csv
- usar spyder para len d.query y nos diga cuantos son spam y cuántos no
- luego vectorizar para saber cuántas palabras se usaron
- no creo que debamos vectorizar en números (54)
- shuf ? train ?
- luego que es un tree

Bitácora semana 2: Seguimos leyendo la problemática, empezamos y pudimos descargar exitosamente la base de datos, seleccionamos la base de datos del video de Gangnam Style, guardamos esa base de datos e intentamos leerla en spyder para empezar a programar, pero no nos fue posible leer la base de datos.



```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]:
In [1]: import pandas as pd
In [2]: d = pd.read_csv("D:/Chapter04/Youtube01-Psy.csv")
Traceback (most recent call last):
```

Después logramos leer el archivo y empezamos a programar, usamos la función tail para comprobar que los últimos datos estén correctamente, después usamos la función query, para contar cuantos comentarios son spam y cuántos no.

```

In [1]: import pandas as pd
In [2]: df= pd.read_csv("Youtube01-Psy.csv")
In [3]: df.tail()
Out[3]:
          COMMENT_ID  ... CLASS
345  z13th1q4yzihf1bll23qxzpjuejterydj  ...    0
346  z13fcn1wfpb5e51xe04chdxakpzgchyaxzo0k  ...    0
347  z130zd5b3titudkoe04ccbeohojxuzppvbg  ...    1
348  z12he50arvrkiv15u04cctawgxzkjfsjcc4  ...    1
349  z13vhvu54u3ewpp5h04ccb4zuoardrmjlyk0k  ...    0

[5 rows x 5 columns]

In [4]: len(df.query("CLASS==1"))
Out[4]: 175

In [5]: len(df.query("CLASS==0"))
Out[5]: 175

In [6]: len(df)
Out[6]: 350

In [7]: from sklearn.feature_extraction.text import CountVectorizer

```

Luego vectorizamos todos los comentarios para contarlos en filas y palabras, obtuvimos que 175 eran spam y 175 no eran spam, la función dvec nos sirvió para saber que tenemos 350 renglones y 1418 palabras .

Bitácora semana 3:

```

In [13]: analyze(d["CONTENT"][349])
Traceback (most recent call last):

  File "C:\Users\gabri\AppData\Local\Temp/ipykernel_64340/209415651.py", line 1, in <module>
    analyze(d["CONTENT"][349])

NameError: name 'analyze' is not defined

```

Inicialmente al momento de utilizar la función analyze nos encontramos un error, pues al momento de utilizar el código no funcionaba debido a que la función analyze no estaba definida.

Con dicha función tendríamos la capacidad de seleccionar un comentario en específico y dividirlo por cada una de las palabras que incluye. De esta forma tendremos la capacidad de identificar las palabras que son spam o no en un comentario.

```

In [19]: pd.analyze(df["CONTENT"][349])
Traceback (most recent call last):
  File "C:\Users\RODOLFO\1\AppData\Local\Temp\ipykernel_10836\1362514469.py",
    line 1, in <module>
      pd.analyze(df["CONTENT"][349])
  File "C:\Users\Rodolfo Aceves\anaconda3\lib\site-packages\pandas\__init__.py", line 244, in __getattr__
    raise AttributeError(f"module 'pandas' has no attribute '{name}'")
AttributeError: module 'pandas' has no attribute 'analyze'

```

Este fue un segundo intento para solucionar nuestro error. Finalmente descubrimos los pasos que debían seguirse para poder realizar la función de manera correcta:

```

In [15]: analyze = vectorizer.build_analyzer()

```

```

In [17]: analyze(df["CONTENT"][349])
Out[17]:
['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',
]

```

Semana 3, parte 2:

```

In [8]: vectorizer = CountVectorizer()

In [9]: dvec = vectorizer.fit_transform(df["CONTENT"])

In [10]: dvec
Out[10]:
<350x1418 sparse matrix of type '<class 'numpy.int64'>'
  with 4354 stored elements in Compressed Sparse Row format>

In [11]: print(df["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 [12]: dfshuf= df.sample(frac=1)

In [13]: df_train = dfshuf[:300]

In [14]: df_test = dfshuf[300:]

In [15]: df_train_att= vectorizer.fit_transform(df_train["CONTENT"])

In [16]: df_test_att= vectorizer.transform(df_test["CONTENT"])

In [17]: df_train_label = df_train["CLASS"]

In [18]: df_test_label = df_test["CLASS"]

```

Luego usamos el featurename para vectorizar las palabras, después de eso usamos la función shuf para revolver todas las palabras, luego usamos las función train y test para ir entrenando al código en realizar el wordbag y que sepa identificar el spam, en una relación 80% train y 20% test.

Pero esto qué significa, los datos se dividen en subconjunto y con el train se pone a prueba el machine learning para ver la eficacia de nuestro proyecto.

Después de usar el random forest classifier y comprobar la confiabilidad del 98% se utiliza una validación de 5 diferentes variables, vamos a utilizar la training data y separarla en diferentes grupos uno de 20% otro de 80% y otro 20% para tener un total de 80% de training data .

```

In [19]: df_train_att
Out[19]:
<300x1280 sparse matrix of type '<class 'numpy.int64'>'
  with 3760 stored elements in Compressed Sparse Row format>

In [20]: df_test_att
Out[20]:
<50x1280 sparse matrix of type '<class 'numpy.int64'>'
  with 456 stored elements in Compressed Sparse Row format>

In [21]: from sklearn.ensemble import RandomForestClassifier

In [22]: clf= RandomForestClassifier(n_estimators=80)

In [23]: clf.fit(df_train_att, df_train_label)
Out[23]: RandomForestClassifier(n_estimators=80)

In [24]: clf.score(df_test_att, df_test_label)
Out[24]: 0.96

In [25]: dfshuf= df.sample(frac=1)

In [26]: from sklearn.metrics import confusion_matrix

In [27]: pred_labels = clf.predict(df_test_att)

```

Nos dimos cuenta que con el `clf.fit`, que el programa tiene un 96% de precisión, esto es un porcentaje bastante alto, hasta que se usó el test y el train y seguimos confirmando el porcentaje.

Semana 4:

Luego usamos las matrices de confusión para comprobar la precisión del programa y este bajó a 95%, lo cual sigue siendo un buen indicador, a continuación, juntamos todas las bases de datos de distintos videos de youtube, en total teníamos 1956 comentarios, donde 1005 comentarios son spam y 951 no.

```
In [28]: confusion_matrix(df_test_label, pred_labels)
Out[28]:
array([[26,  0],
       [ 2, 22]], dtype=int64)

In [29]: from sklearn.model_selection import cross_val_score

In [30]: scores= cross_val_score(clf, df_train_att, df_train_label, cv=5)

In [31]: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
Accuracy: 0.95 (+/- 0.04)

In [32]: df= pd.concat([pd.read_csv("Youtube01-PSY.csv"), pd.read_csv("Youtube02-KatyPerry.csv"), pd.read_csv("Youtube03-LMFAO.csv"), pd.read_csv("Youtube04-Eminem.csv"), pd.read_csv("Youtube05-Shakira.csv")])

In [33]: len(df)
Out[33]: 1956

In [34]: len(df.query("CLASS == 1"))
Out[34]: 1005

In [35]: len(df.query("CLASS == 0"))
Out[35]: 951
```

Volvimos a usar el shuffle para tener una mezcla de los datos, así pudimos usar pipeline para juntar las funciones que ya se habían usado y de esta forma, podemos decir que se juntaron todos los pasos en un renglón.

```
In [37]: dfshuf=df.sample(frac=1)

In [38]: df_content = dfshuf["CONTENT"]

In [39]: df_label = dfshuf["CLASS"]

In [40]: from sklearn.pipeline import Pipeline, make_pipeline

In [41]: pipeline = Pipeline([("bag-of-words", CountVectorizer()), ("random forest", RandomForestClassifier()),])

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

In [43]: make_pipeline(CountVectorizer(), RandomForestClassifier())
Out[43]:
Pipeline(steps=[('countvectorizer', CountVectorizer()),
                 ('randomforestclassifier', RandomForestClassifier())])

In [44]: pipeline.fit(df_content[:1500],df_label[:1500])
Out[44]:
Pipeline(steps=[('bag-of-words', CountVectorizer()),
                 ('random forest', RandomForestClassifier())])
```

Después pusimos a prueba el código y los resultados fueron correctos, nos marcaba cuáles eran spam y cuáles no, incluso nosotros escribiendo comentarios nos arrojaba los resultados correctos.


```

In [47]: pipeline.predict(["what a neat video"])
Out[47]: array([0], dtype=int64)

In [48]: pipeline.predict(["plz subscribe to my channel"])
Out[48]: array([1], dtype=int64)

In [49]: pipeline.predict(["awesome song"])
Out[49]: array([0], dtype=int64)

In [50]: pipeline.predict(["can you give me money plz"])
Out[50]: array([1], dtype=int64)

In [51]: scores= cross_val_score(pipeline, df_content, df_label, cv=5)

In [52]: print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
Accuracy: 0.96 (+/- 0.01)

```

Volvimos a usar pipeline para volver a medir la precisión y nos arrojó 96%, aumentando la precisión del programa.

```

In [54]: from sklearn.feature_extraction.text import TfidfTransformer

In [55]: pipeline2 = make_pipeline(CountVectorizer(), TfidfTransformer(norm=None), RandomForestClassifier())

In [56]: scores= cross_val_score(pipeline2, df_content, df_label, cv=5)

In [57]: print ("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
Accuracy: 0.96 (+/- 0.02)

In [58]: pipeline2.steps
Out[58]:
[('countvectorizer', CountVectorizer()),
 ('tfidftransformer', TfidfTransformer(norm=None)),
 ('randomforestclassifier', RandomForestClassifier())]

In [59]: parameters= { "countvectorizer_max_features": (None, 1000, 2000), "countvectorizer_ngram_range": ((1,1), (1,2)),
"countvectorizer_stop_words": ("english", None), "tfidftransformer_use_idf": (True, False), "randomforestclassifier_n_estimators":
(20, 50, 100)}

In [60]: from sklearn.model_selection import GridSearchCV

In [61]: grid_search = GridSearchCV(pipeline2, parameters, n_jobs=-1, verbose=1)

```

Grid nos sirvió para los últimos pasos, donde te da un resumen de las bases de datos que se juntaron.

```

In [62]: grid_search.fit(df_content, df_label)
Fitting 5 folds for each of 72 candidates, totalling 360 fits
RemoteTraceback:

```

Los últimos pasos del código, marcaron error, pero nos dimos cuenta que no era totalmente necesario para el programa, ya que ya sabíamos cómo sacar el spam y el no spam, entonces el objetivo se cumplió.

```

In [63]: print("Best score: %0.3f" % grid_search.best_score_)
Traceback (most recent call last):

  File "C:\Users\RODOLFO~1\AppData\Local\Temp\ipykernel_13100\3604662991.py", line 1, in <module>
    print("Best score: %0.3f" % grid_search.best_score_)
AttributeError: 'GridSearchCV' object has no attribute 'best_score_'

In [64]: print("Best parameters set:") best_parameters = grid_search.best_estimator_.get_params()
File "C:\Users\RODOLFO~1\AppData\Local\Temp\ipykernel_13100\199710529.py", line 1
    print("Best parameters set:") best_parameters = grid_search.best_estimator_.get_params()
                                     ^
SyntaxError: invalid syntax

In [65]: print("Best parameters set:")
...: best_parameters = grid_search.best_estimator_.get_params()
Best parameters set:
Traceback (most recent call last):

  File "C:\Users\RODOLFO~1\AppData\Local\Temp\ipykernel_13100\868546287.py", line 2, in <module>
    best_parameters = grid_search.best_estimator_.get_params()
AttributeError: 'GridSearchCV' object has no attribute 'best_estimator_'

```

Mariano: Durante todo el curso se tomó como prioridad el entendimiento de los datos y creo que eso fue lo que más aprendí en el curso, no solo a programar en modo robot, si no a saber qué significa los datos que arroja el programa saber que la confiabilidad de un modelo y saber interpretar el mismo y saber que conviene hacer en ciertas ocasiones y que datos son o no irrelevantes para hacer un modelo aún más preciso.

Gabriel: Gracias a la realización de esta actividad en equipo aprendí nuevas librerías y conceptos de la Ciencia de Datos además de crear un código con utilidad a la vida real y pueda ser de guía para futuros proyectos. Las enseñanzas de la clase a pesar de no estar directamente relacionadas con mi carrera a estudiar, en mi opinión vienen siendo aprendizajes valiosos que te permiten conocer aunque sea lo básico de programación y ciencia de datos. Lo veo con utilidad para el futuro ya que uno en negocios trabaja con números y datos de tu empresa o de otra por lo que contar con este tipo de conocimiento te da la capacidad de tratar los datos a tu favor o pedirle a un experto que lo haga teniendo la capacidad de darle instrucciones claras y precisas. Otro aprendizaje viene siendo la manera en la que llevamos el trabajo a cabo, pues la procrastinación inicial nos arrastró mucho no obstante con esta experiencia me llevo el aprendizaje de trabajar todo desde un inicio para optimizar tus tiempos y poder realizar hasta más de lo que se pide.

Ángel: En la duración del curso pude tener diferentes aprendizajes, desde para que sirve el entendimiento de datos y diferentes conocimientos de programación, algo que en lo personal se me dificulta demasiado pero con el seguimiento del profesor y la ayuda de algunos compañeros, logre realmente completar mi modelo de análisis de nutrientes y entenderlo. Como conclusión, considero que los conocimientos obtenidos durante el curso me piden servir en el ámbito de los negocios, ya sea para entender el comportamiento de algún producto o la aceptación que tiene en el mercado.

Alejandro: A lo largo de este semestre pude aprender varias cosas sobre la ciencia de datos, ya que era un concepto que yo no conocía para nada y me he dado cuenta de toda la utilidad que tiene en la vida diaria, tan solo con este ejercicio me puedo dar cuenta que incluso para redes sociales puede servir la ciencia de datos y puedes detectar fenómenos para poder intentar predecir cosas o tan solo interpretar los datos que tienes en cualquier lugar, Esto es lo que me dejó con mayor satisfacción la ciencia de datos, que realmente cualquier base de datos o cualquier lugar donde tienes datos puedes interpretarlo para poder tomar decisiones o analizarlos y entender los fenómenos que están en el entorno y creo que puedo relacionar todos estos aprendizajes con el área que voy a estudiar que es negocios, porque creo que lo más importante en los negocios son los datos para poder saber a que público llegar y poder predecir problemas que se pueden presentar, por lo cual creo que la ciencia de datos me puede ayudar bastante.