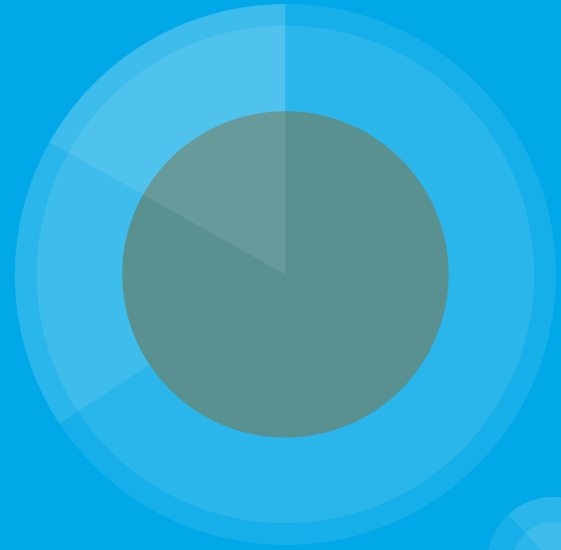


Proyecto Machine Learning - Bitácora

Andrés Arroccena Perches - A01571430
Fernanda Jazmin Narvaez Andrade - A00572827
Octavio Vizuet Hanafuji - A00573944







Semana 1

(ppt 4-15)




Fecha: 01/11/22

Acciones realizadas: Selección de proyecto, familiarización del mismo y organización de trabajo en equipo.

Fecha: 04/11/22

Acciones Realizadas: Se realiza el contenido conceptual necesario y se investiga el funcionamiento del spam y de la sección de comentarios de youtube.



**Propósito del proyecto:
Detectar en la sección de
comentarios de la
plataforma Youtube como
spam o no spam**



SPAM

¿Qué es? ¿Cómo funciona? ¿Por qué se utiliza?
¿Por qué en youtube?

¿Qué es?

El spam es conocido como los correos basura o mensajes basura, es una forma de comunicación no solicitada que se envía de manera masiva.



¿Por qué se utiliza?



Por lo general lo usan para mostrar cosas en venta, mostrar ideas, pensamientos, estafar a gente o enviar virus a dispositivos.

¿Cómo funciona?

Normalmente se maneja por correo electrónico a través de botnets, grandes redes de dispositivos "zombie" infectados. Cuando es por mensaje se copia e imprime de manera repetitiva en diferentes “chats” o bandejas de entrada.





Youtube.

¿Qué es? ¿Cómo funciona? Su sección de comentarios



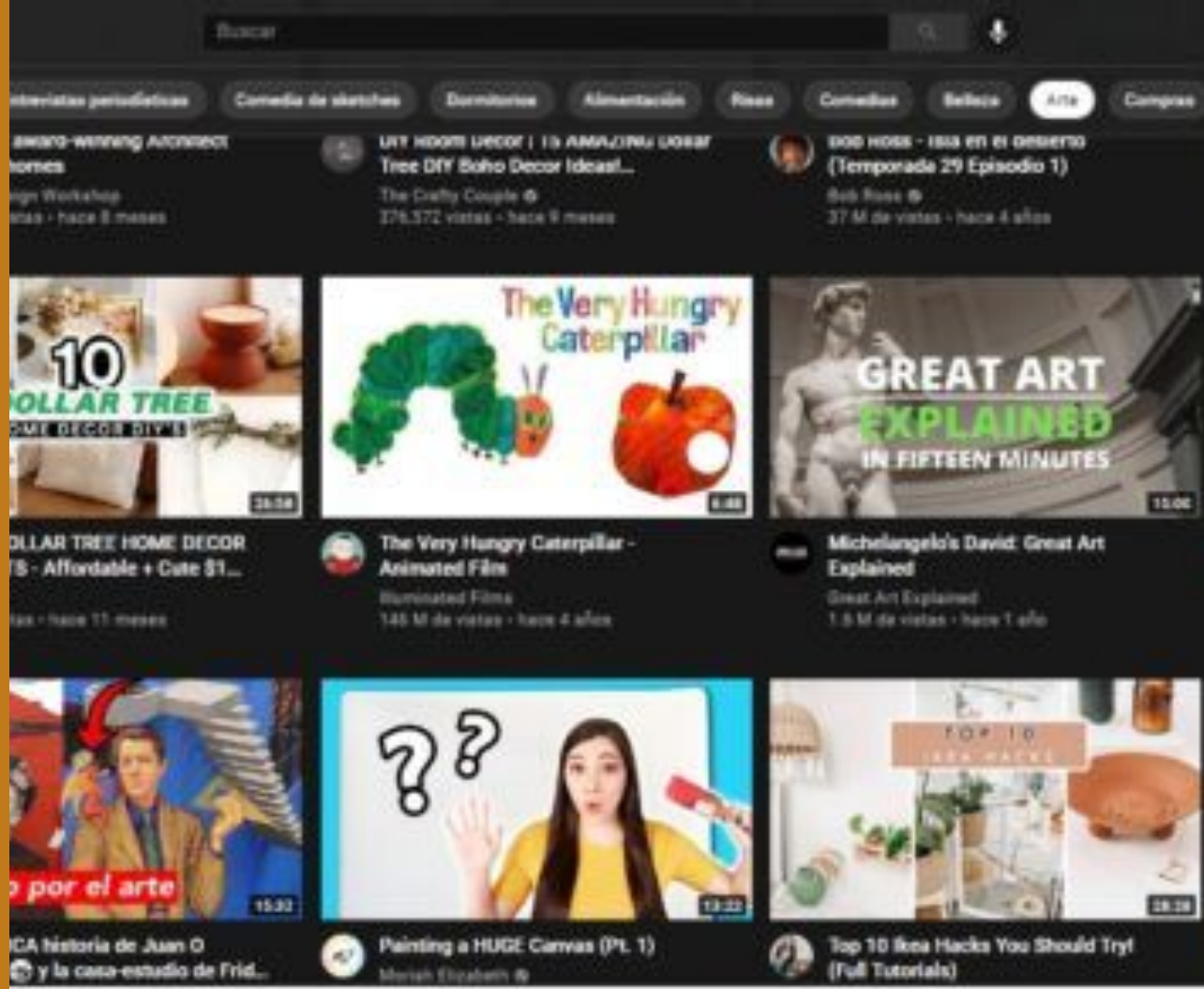
¿Qué es?

Youtube es una plataforma la cual se utiliza para ver videos ya sea de entretenimiento, académico, de aprendizaje o laboral.



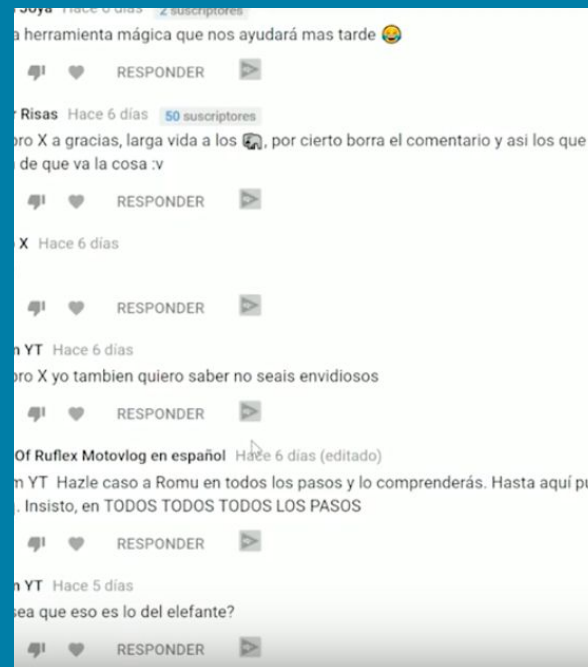
¿Cómo funciona?

El funcionamiento de Youtube es muy simple, se ingresa a la plataforma de manera gratuita en el que sale un buscador donde puedes teclear algo y te saldrán videos del tema que buscaste.



Su sección de comentarios

En cada video hay una sección de comentarios donde los usuarios podrán escribir cualquier cosa y leer todos los comentarios de manera anónima o con algún perfil personal.



Relación de spam y youtube

¿Por qué en youtube? Por lo general se utiliza el spam en youtube ya que en esta plataforma hay mucha gente y el mensaje le puede llegar a muchas más personas, al ser de manera anónima no hay repercusiones graves en el individuo.

Tipos de Spam Comentarios en Youtube

Spam de incentivación: Contenido en el que se venden métricas de participación, como vistas, “me gusta”, comentarios o cualquier otra métrica de YouTube; también puede incluir contenido cuyo único propósito es aumentar la cantidad de suscriptores, vistas y otras métricas, por ejemplo, el ofrecimiento de suscribirte al canal de otro creador a cambio de que este se suscriba al tuyo (esta práctica también se conoce como “sub4sub”)

Comentarios spam: Comentarios cuyo único propósito es recopilar información personal de los usuarios o dirigirlos fuera de YouTube mediante un engaño.

2018 YOUTUBE

Durante dicho trimestre se eliminaron más de 224 millones de comentarios, la mayoría referidos a spam.





Semana 2

(ppt 18-25)



Fecha: 7/11/2022 - 11/11/2022

Acciones realizadas:

- Descargar y analizar la base de datos, cargarla a Spyder y asegurar que es correcta y funcional.
- Decretar e investigar las funciones a utilizar.
- Definir las librerías a importar.
- Determinar el entorno del algoritmo y su teoría.



Descargamos la base de datos brindada por el libro de “Python Artificial Intelligence Projects for Beginners” escrito por Joshua Eckroth. Usaremos una base con aproximadamente 2000 comentarios

YouTube Spam Collection Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: It is a public set of comments collected for spam research. It has five datasets composed by 1,956 real messages extracted from 1 period.

Data Set Characteristics:	Text	Number of Instances:	1956	Area:	Computer
Attribute Characteristics:	N/A	Number of Attributes:	5	Date Donated	2017-03-26
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	115832

Source:

This corpus has been collected using the YouTube Data API v3.

Data Set Information:

The table below lists the datasets, the YouTube video ID, the amount of samples in each class and the total number of samples per dataset.

```
Dataset --- YouTube ID -- # Spam - # Ham - Total
Psy ----- 9bZkp7q19f0 --- 175 --- 175 --- 350
KatyPerry - CevxZvSJLk8 --- 175 --- 175 --- 350
LMFAO ----- KQ6zr6kCPj8 --- 236 --- 202 --- 438
Eminem ----- uelHwf8o7_U --- 245 --- 203 --- 448
Shakira --- pRpeEdMmmQ0 --- 174 --- 196 --- 370
```

```
File ~\anaconda3\lib\site-packages\pandas\io\parsers\readers.py:1217 in _make_engine
    self.handles = get_handle( # type: ignore[call-overload]

File ~\anaconda3\lib\site-packages\pandas\io\common.py:789 in get_handle
    handle = open(

FileNotFoundError: [Errno 2] No such file or directory: 'Youtube01-Psy.csv'

In [5]: df = pd.read_csv("Youtube01-Psy.csv")

In [6]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   COMMENT_ID  350 non-null    object
1   AUTHOR      350 non-null    object
2   DATE        350 non-null    object
3   CONTENT     350 non-null    object
4   CLASS       350 non-null    int64
dtypes: int64(1), object(4)
memory usage: 13.8+ KB
```

Funciones

- query

```
In [3]: len(d.query('CLASS == 1'))  
Out[3]: 175  
  
In [4]: len(d.query('CLASS == 0'))  
Out[4]: 175  
  
In [5]: len(d)  
Out[5]: 350
```

Ejecuta una consulta sobre los datos con el lenguaje de consultas de la API de visualización de Google.

- Tail

```
In [2]: d.tail()  
Out[2]:
```

	COMMENT_ID	DATE	CONTENT	CLASS
345	z13th1q4yzihf1bl23qzpjoujterjdj	2014-11-14T13:27:52	How can this have 2 billion views when there's...	0
346	z13fctn1wfpb5e51xe04chdxakpzgchyaxzo0k	2014-11-14T13:28:08	I don't now why I'm watching this in 2014	0
347	z130zd5b3ttudkoe04ccbeohojxuzppvbg	2015-05-23T13:04:32	subscribe to me for call of duty vids and give...	1
348	z12he50arvrkivf5u04cctawgxzkfajcc4	2015-06-05T14:14:48	hi guys please my android photo editor downlo...	1
349	z13vhu54u3ewpp5h04ccb4zuoardmlyk0k	2015-06-05T18:05:16	The first billion viewed this because they tho...	0

Retorna las últimas filas de un objeto basado en su posición. Para verificar datos de manera veloz.

- **.fit_transform**

```
In [6]: from sklearn.feature_extraction.text import CountVectorizer  
vectorizer = CountVectorizer()  
  
In [7]: dvec = vectorizer.fit_transform(d['CONTENT'])
```

This performed in two different steps. First comes the fit step, where it discovers which words are present in the dataset, and second is the transform step, which gives you the bag of words matrix for those phrases. The result obtained in that matrix is 350 rows by 1,418 columns:

```
In [7]: dvec = vectorizer.fit_transform(d['CONTENT'])  
  
In [8]: dvec  
Out[8]: <350x1418 sparse matrix of type '<class 'numpy.int64'>'  
         with 4354 stored elements in Compressed Sparse Row format>
```

Se utiliza para escalar los datos de entrenamiento y para aprender los parámetros de escalado de esos datos.

- **pipeline.predict**

Genera un modelo para cualquier problema de clasificación o regresión.

```
In [31]: pipeline.predict(["what a neat video!"])
```

```
Out[31]: array([0], dtype=int64)
```

As seen, it's detected correctly; but what about the following comment:

```
In [32]: pipeline.predict(["plz subscribe to my channel"])
```

```
Out[32]: array([1], dtype=int64)
```

To overcome this and deploy this classifier into an environment and predict whether it is a spam or not when someone types a new comment. We will use our pipeline to figure out how accurate our cross-validation was. We find in this case that the average accuracy was about 94:

```
In [33]: scores = cross_val_score(pipeline, d_content, d_label, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.94 (+/- 0.02)
```

```
In [11]: vectorizer.get_feature_names()
```

```
Out[11]: ['00',  
          '000',  
          '02',  
          '034',  
          '05',  
          '08',  
          '10',  
          '100',  
          '100000415527985',  
          '10200253113705769',  
          '1030',  
          '1073741828',  
          '11',  
          '1111',  
          '112720997191206369631',  
          '12',  
          '123',  
          '124',  
          '124923004']
```

- **.get_feature_names**

Para averiguar qué palabra encontró el conjunto de datos después de vectorizar.

```
In [12]: dshuf = d.sample(frac=1)
```

Elige los
comentarios al
azar.

- **.sample**

Devuelve de una lista de elementos un determinado número de elementos diferentes elegidos al azar.

Librerías que estamos usando

- Pandas
- Sklearn
- Zipfile
- Confusion matrix
- Google.colab
- Random Forest Classifier

Teoría y Entorno de algoritmo



Base de datos, data set

Nuestro conjunto de datos contiene 2000 comentarios sacados de videos populares de youtube. A lado de cada comentario, en otra columna, se muestra un 0 o un 1. Dependiendo de si el comentario es spam o no.

Bag of Words

Una bolsa de palabras es una estructura de datos que realiza un seguimiento de los objetos como lo hace una matriz o una lista.

Vamos a convertir nuestros comentarios en una bolsa de palabras, convirtiendo las frases u oraciones y contando la cantidad de veces que aparece una palabra similar.

La técnica de la bolsa de palabras realmente se conoce como 'CountVectorizer'. (cuenta las veces que aparece una palabra en un vector)

En futuras sesiones se implementará al código.



Random Forest

Es una algoritmo y técnica de clasificación.

Son extensiones de árboles de decisión, es un estimador que ajusta una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobreajuste.

Nosotros convertiremos nuestros datos en 80 diferentes árboles de decisión.

```
from sklearn.ensemble import  
RandomForestClassifier
```

```
RFC =  
RandomForestClassifier(n_estimators  
= 80, random_state = 0)
```



Semana 3

(ppt 27- 31)

Fecha: 14/11/2022

Acciones realizadas:

- Progreso en código.

Fecha: 18/11/2022

Acciones realizadas:

- Seguimiento del código
- Explicación del funcionamiento por línea de código y organización del mismo.

```

In [4]: df.tail()
Out[4]:

```

	COMMENT_ID	...	CLASS
345	z13th1q4yzihf1b1123qxzpjuejterydj	...	0
346	z13fcn1wfpb5e51xe04chdxakpzgchyaxzo0k	...	0
347	z130zd5b3titudkoe04ccbeohojxuzppvbg	...	1
348	z12he50arvrkiv15u04cctawgxzkjfsjcc4	...	1
349	z13vhvu54u3ewpp5h04ccb4zuoardrmjlyk0k	...	0

```

[5 rows x 5 columns]

In [5]: len(df.query('CLASS == 1'))
Out[5]: 175

In [6]: len(df.query('CLASS == 0'))
Out[6]: 175

In [7]: len(df)
Out[7]: 350

In [8]: from sklearn.feature_extraction.text import CountVectorizer
...: vectorizer = CountVectorizer()

In [9]: dfvec = vectorizer.fit_transform(df['CONTENT'])

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

```

Empezaremos con un solo conjunto de datos de la base, que está dividido en otros cuatro conjuntos de datos.

La columna de contenido contiene los comentarios y la clase. Cada columna contiene los valores 1 o 0 para spam o no spam.

```

In [4]: df.tail()
Out[4]:

```

	COMMENT_ID	...	CLASS
345	z13th1q4yzihf1b1l23qxzpjuejterydj	...	0
346	z13fcn1wfpb5e51xe04chdxakpzgchyaxzo0k	...	0
347	z130zd5b3titudkoe04ccbeohojxuzppvbq	...	1
348	z12he50arvrkiv15u04cctawgxzkjfsjcc4	...	1
349	z13vhvu54u3ewpp5h04ccb4zuoardrmjlyk0k	...	0

```

[5 rows x 5 columns]

In [5]: len(df.query('CLASS == 1'))
Out[5]: 175

In [6]: len(df.query('CLASS == 0'))
Out[6]: 175

In [7]: len(df)
Out[7]: 350

In [8]: from sklearn.feature_extraction.text import CountVectorizer
...: vectorizer = CountVectorizer()

In [9]: dfvec = vectorizer.fit_transform(df['CONTENT'])

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

```

En esta sección de código se realiza:

Línea 4,5,7:

- Un recuento de los números de datos que hay para CLASS 1 y 0. Función Len.
- Query: función integrada con pandas que permite realizar una búsqueda dentro de un conjunto de valores.

Línea 8:

- Se manda a llamar, se importa, Count Vectorizer. De scikit-learn.
- Se inicializa un objeto, variable, donde se crea el vector con CountVectorizer().,

Línea 9:

- .fit_trandorm: cuenta, analiza, transforma y procesa el contenido de la columna CONTENT.

Línea 10:

- Se muestran los resultados de la función anterior. El resultado obtenido es una matriz de 350 filas (comentarios) por 1418 columnas (palabras diferentes).

```

In [20]: dfvec= vectorizer.fit_transform(df['CONTENT'])

In [21]: dfvec
Out[21]:
<350x1418 sparse matrix of type '<class 'numpy.int64'>'
  with 4354 stored elements in Compressed Sparse Row format>

In [22]: lista = (df['CONTENT'][340])

In [23]: print(df['CONTENT'][340])
Please do buy these new Christmas shirts! You can buy at any time before  December 4th and they are sold worldwide! Don't miss out:
http://teespring.com/treechristmas

In [24]: print(lista.split())
['Please', 'do', 'buy', 'these', 'new', 'Christmas', 'shirts!', 'You', 'can', 'buy', 'at', 'any', 'time', 'before', 'December', '4th',
'and', 'they', 'are', 'sold', 'worldwide!', "Don't", 'miss', 'out:', 'http://teespring.com/treechristmas\ufe0f']

```

En esta sección de código se realiza:

Línea 22 y 23:

- Se guarda el comentario 340 en la variable lista
- Se imprime el comentario 340

Línea 24:

- En sustitución de analyze, utilizamos la función split para dividir y mostrar las palabras en la consola.

```
In [22]: vectorizer.get_feature_names()
```

```
'gabriel',  
'game',  
'games',  
'gamestop',  
'gaming',  
'ganga',  
'gangman',  
'gangnam',  
'gangnamstyle',  
'gatti',  
'gay',  
'gbphotographygb',  
'gcmforex',  
'get',  
'gets',  
'getting',  
'ghost',  
'gift',  
'girl',  
'girls',  
'give',  
'giveaways',  
'giver',  
'glasses',  
'go',  
'goal',  
'gofundme',  
'going'
```

```
In [11]: vectorizer.get_feature_names()
```

```
Out[11]: ['00',  
'000',  
'02',  
'034',  
'05',  
'08',  
'10',  
'100',  
'100000415527985',  
'10200253113705769',  
'1030',  
'1073741828',  
'11',  
'1111',  
'112720997191206369631',  
'12',  
'123',  
'124',  
'124923004',
```

En esta sección de código se realiza:

Línea 22:

- Con la función `.get_feature_names()` se mostrarán los nombres de características seleccionados.
- Esta matriz se ordenará por índice.

El libro nos menciona que con esta función se debería mostrar el resultado de vectores numéricos, lo que se realizó con la función `.fit_transform`.

Buscamos otras alternativas como:

- `get_feature_count`
- `get_num_feature_count`

Example one
Example two

and	back	channel	grow	guys	help	i	me	my	please	plz	subscribe	to	xx
1	1	1	0	0	0	1	0	1	0	1	2	1	1
0	0	1	1	2	1	0	1	1	1	0	1	1	0

```

In [13]: dfshuf = df.sample(frac = 1)

In [14]: df_train = dfshuf[:300]
...: df_test = dfshuf[300:]
...: df_train_att = vectorizer.fit_transform(df_train['CONTENT'])
...: df_test_att = vectorizer.transform(df_test['CONTENT'])
...: df_train_label = df_train['CLASS']
...: df_test_label = df_test['CLASS']

In [15]: df_train_att
Out[15]:
<300x1281 sparse matrix of type '<class 'numpy.int64'>'
  with 3702 stored elements in Compressed Sparse Row format>

In [16]: df_test_att
Out[16]:
<50x1281 sparse matrix of type '<class 'numpy.int64'>'
  with 514 stored elements in Compressed Sparse Row format>

```

- Se dividen y guardan 300 valores en df_train
- Se dividen y guardan 50 valores en df_test

Nuestro conjunto de datos de entrenamiento se enfoca en analizar, transformar, aprender y crear la matriz.

El conjunto de pruebas no va a realizar este entrenamiento, ya que se busca que no tenga nuevas palabras a comparación del conjunto de entrenamiento.

- Se les adjunta una etiqueta

Línea 15y16:

- Los resultados fueron para el conjunto de entrenamiento 300 filas y 1287 columnas , y el conjunto de prueba tiene 50 filas, pero tenemos las mismas 1287 columnas. (filas comentarios, columnas-palabras)

En esta sección de código se realiza:

Línea 13:

- Con el comando .sample se mezcla el conjunto de datos y se guarda en la variable dfshuf.
- Es necesario mezclarlo para tener reproducibilidad (capacidad de un experimento de ser reproducido) y retener relaciones lógicas entre columnas.

Línea 14:

Dividimos nuestro conjunto de datos entre entrenamiento y prueba.



Semana 4

```
In [17]: from sklearn.ensemble import RandomForestClassifier
...: clf = RandomForestClassifier(n_estimators = 80)

In [18]: clf.fit(df_train_att, df_train_label)
Out[18]: RandomForestClassifier(n_estimators=80)

In [19]: clf.score(df_test_att, df_test_label)
Out[19]: 0.96
```

Fecha: 22/11/2022

Acciones realizadas:

- Se finaliza el código.
- Seguimiento de la explicación del funcionamiento del código por línea.
- Investigación Teórica de la matriz de confusión respecto a algoritmo Random Forest

Fecha: 25/11/2022

Acciones realizadas:

- Organización y cierre de Bitácora.
- Explicación final y resumida del código restante.
- Conclusiones personales del proyecto.
- Elaboración de documento con código y bibliografía


```
In [17]: from sklearn.ensemble import RandomForestClassifier
...: clf = RandomForestClassifier(n_estimators = 80)

In [18]: clf.fit(df_train_att, df_train_label)
Out[18]: RandomForestClassifier(n_estimators=80)

In [19]: clf.score(df_test_att, df_test_label)
Out[19]: 0.96
```

En esta sección de código se realiza:

Línea 17:

- Se importa, RandomForestClassifier... De scikit-learn.ensemble
- Se crea modelo para el random forest. Se inicializa un objeto donde se crea el vector. Y se controla el número de submuestras a crear con **n_estimators**. Por determinado se puede de 10 a 100, se eligió 80 para ir a la par del libro.

Línea 18:

- .fit= Ajusta el modelo de random forest con los valores del subconjunto de vectores en entrenamiento

Línea 19:

- .score = Mide la precisión del modelo con respecto a los datos de entrenamiento. Nuestro modelo da un resultado alto de 0.96.

A continuación se realizará una validación del puntaje de precisión con una matriz de confusión...

Matriz de confusión: Herramienta que permite visualizar el desempeño de un algoritmo (permite ver qué tipos de aciertos y errores está teniendo el modelo a la hora de pasar por el proceso de aprendizaje con los datos).

En nuestro caso: mostrará qué tan bien el árbol de decisiones separa correctamente las clases utilizando métricas:

- Tasa de verdaderos positivos
- Tasa de falsos positivos
- Tasa de falsos negativos
- Tasa de verdaderos negativos

```
In [21]: from sklearn.metrics import confusion_matrix
...: pred_labels = clf.predict(df_test_att)
...: confusion_matrix(df_test_label, pred_labels)
Out[21]:
array([[31,  0],
       [ 2, 17]], dtype=int64)
```

En esta sección de código se realiza:

Línea 21:

- Se importa, confusion_matrix... De scikit-learn.metrics
- Predecimos las etiquetas de los valores de los datos sobre el modelo de submuestras.
- Se crea la matriz de confusión con el conjunto de prueba, y la predicción realizada anteriormente.

Tenemos un total de 48 predicciones correctas de 50. Esa es una precisión bastante buena.

Código del Proyecto

URL de documento que contiene el código finalizado y
Completo

Resumen Explicación Código Restante

Validación cruzada: Aprendizaje automático que divide los datos en varios subconjuntos, crea un modelo en cada uno y devuelve un conjunto de estadísticas de precisión.

PipeLine: Función de scikit-learn que reúne a dos o más pasos para que todos los pasos se traten como uno solo. Una canalización.

- Después de realizar una predicción con una matriz de confusión realizamos una validación cruzada con cuatro divisiones diferentes. Obteniendo una predicción de 96% de precisión.
- Cargamos toda la base de datos, en total 2000 filas de comentarios.
- Se mezclan los datos y separamos contenido y clase.
- Usando Pipeline construimos una instrucción en conjunto: primero bolsa de palabras, seguido de countVectorizer y finalmente el random classifier. (Instrucciones ya explicadas anteriormente). De esta predicción de entrenamiento se obtuvo un 96% de precisión.
- Con la canalización podemos predecir si un comentario es spam o no. Agregamos dos comentarios y se predicen de manera correcta.



Conclusiones



Principal Aprendizaje

Dar las instrucciones y funciones a una máquina para que esta pueda realizar predicciones.

Un sin fin de entornos de algoritmos, especializadas, enfocadas y detalladas en distintos campos. Cada una de ellas realiza modelos distintos.

Leer Código. Relacionar funciones. Librerías. El uso de código abierto.



Para poder realizar el proyecto tuvimos que aprender a trabajar en equipo, ya que al dividirse la carga pudimos sacar un mayor rendimiento y un mejor proyecto. Tuvimos que familiarizarnos con spyder y prestar atención a la información dada anteriormente como código de ejemplo.

Octavio

Para poder realizar adecuadamente el proyecto tuvimos que familiarizarnos con spyder para así poder empezar a codificar, tuvimos que tener una buena organización y prestar mucha atención a los datos dados por el profesor, el hacer el código fue lo más laborioso pero al trabajar en equipo nos dividimos la carga y fue un esfuerzo dividido.

Fernanda Jazmin Narvaez Andrade

Por medio de la elaboración de este proyecto logré desarrollar más mis habilidades de deducción y matemática. Aprendí acerca de machine learning y me interesó mucho la capacidad de la inteligencia artificial o computadoras de aprender tareas específicas y lo más importante adaptarse de forma independiente a nuevos datos. Así mismo el sin fin de algoritmos que existen para la predicción y precisión de los datos.

Disfruté la realización de este proyecto. Realmente las herramientas vistas pueden dar resultados increíbles, herramientas que serán útiles en mi futura carrera.

Referencias

Akshay Laddha. (2020, August 30). Detecting spam comments on YouTube using Machine Learning. Retrieved November 25, 2022, from Medium website:

https://medium.com/@akshmahesh/detecting-spam-comments-on-youtube-using-machine-learning_-948d54f47b3

¿Correo basura o spam? Definición de mensajes no deseados. (s. f.). ESET. Recuperado 3 de noviembre de 2022, de <https://www.eset.com/es/caracteristicas/spam/>

likebupt. (2022, September 26). Modelo de validación cruzada: referencia del componente - Azure Machine Learning. Retrieved November 25, 2022, from Microsoft.com website:

<https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/cross-validate-model>

pandas.DataFrame.info — pandas 1.5.2 documentation. (2022). Retrieved November 25, 2022, from Pydata.org website:

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.info.html>

scikit-learn: machine learning in Python — scikit-learn 1.1.3 documentation. (2022). Retrieved November 25, 2022, from Scikit-learn.org website: <https://scikit-learn.org/stable/index.html#>

Python artificial intelligence projects for beginners : get up and running with artificial intelligence using 8 smart and exciting AI applications /