UNIVERSIDAD AUSTRAL | INGENIERÍA

MAESTRÍA EN EXPLOTACIÓN DE DATOS Y GESTIÓN DEL CONOCIMIENTO

WEB MINING

TP 1

ALUMNOS: OTRINO, FACUNDO DAMIÁN

ROJAS, MARIANO ARTURO

VAILLARD, LEANDRO CARLOS

FECHA: 29 DE SEPTIEMBRE DE 2021

Índice

Consigna

Su objetivo es entrenar un clasificador que sea capaz de diferenciar página con los “El Mundo”, “Economía” y “Sociedad”. Para esto deberá descargar usando un crawler a páginas de la versión online del diario Página 12, y entrenar un clasificador, y generar los mejores resultados de cross-validation que pueda conseguir, comparado con sus compañeros. Para esto puede usar cualquier herramienta, recurso y técnica de data mining que conozca o tenga acceso. A continuación, se sugieren las herramientas y pasos utilizando Jobo para descargar las noticias y Python para clasificar.

Entregables:

1. Los scripts de Python (o cualquier otro lenguaje) que haya utilizado.
2. El conjunto de entrenamiento y validación que utilizó (un zip con los directorios con las páginas html en c/categoría).
3. Un documento conteniendo:
    3.1. La descripción de lo realizado para obtener el mejor modelo (algoritmos, parámetros y transformaciones de datos) y que la solución sea generalizable (o sea, que los resultados no sean simplemente overfitting), junto con los resultados de cualquier otra métrica y análisis que haya realizado.
    3.2. La evaluación del mejor resultado de 2 maneras:

        a. Las Matrices de Confusión. ¿observa algo en particular? ¿Hay alguna clase más difícil que las otras? ¿Hay pares de secciones en que el clasificador se confunde más hacia un lado que hacia el otro?

        b. Una división temporal donde entrena con las noticias hasta una cierta fecha y testea con las noticias luego de esa fecha. Claramente el período de test tiene que ser un intervalo temporal que represente la totalidad del intervalo en que descargó noticias.

¿Hay diferencia en los resultados entre ambas maneras de evaluar?

Exploración y Entendimiento de la Página Web

Para desarrollar la consigna planteada, se pueden identificar 4 etapas, las cuales se mencionan a continuación:

- Exploración y entendimiento de la página web
- Desarrollo del *scrapper*
- Vectorización, unificación del corpus y tratamiento de *features*
- Entrenamiento y validación del modelo

Los archivos para llevar a cabo el presente trabajo y desarrollo de cada una de las etapas se encuentran en el siguiente repositorio de GitHub: https://github.com/fotrino/webmining/tree/main/pagina12.

Inspección Exploratoria

Como paso previo al desarrollo del código que permitirá extraer el texto de los artículos correspondientes a las tres secciones del sitio de Página12 (https://www.pagina12.com.ar/) se llevó a cabo una inspección exploratoria de las páginas para conocer su estructura. Para ello se llevaron a cabo los siguientes pasos:

1. Navegar hacia las tres secciones del diario e inspeccionar el código fuente para poder conocer de antemano la estructura de las páginas y así poder desarrollar el *web crawler*. Este proceso se hizo por medio de la herramienta "Inspect" disponible en Google Chrome. Dado que el código es análogo para las tres secciones, se optó por empezar por una de ellas y replicar el proceso para las demás. Las páginas de las tres secciones del diario son:
    a. Economía: https://www.pagina12.com.ar/secciones/economia?page=0
    b. El Mundo: https://www.pagina12.com.ar/secciones/el-mundo?page=0
    c. Sociedad: https://www.pagina12.com.ar/secciones/sociedad?page=0
2. Se analizaron las tres secciones con el fin de determinar qué cantidad de artículos corresponde a un año de noticias. El resultado fue que alrededor de trescientas (300) páginas de cada sección, lo que representa tres mil trescientos (3.300) artículos, siendo la cantidad aproximada de artículos utilizadas por cada una de las secciones. Cabe destacar que se optó por usar la misma cantidad de artículos por sección en lugar de utilizar la misma fecha de inicio y fin del período, con el fin de disponer un dataset balanceado en cantidad de noticias.

Luego de finalizado la inspección exploratoria se inicia el proceso de generación de los archivos que permitirán extraer la información requerida.

Desarrollo Del *Scrapper*

Descripción de los Archivos

A continuación, se brinda una breve descripción de cada uno de los archivos que permiten resolver el presente trabajo práctico.

Archivo links.py

Este archivo contiene la función que permite extraer las direcciones de las páginas web correspondientes a cada artículo.

Se optó por utilizar la biblioteca *scrapy* para el desarrollo del código. A continuación, se brinda una imagen del código para su posterior explicación:

```
1   import scrapy
2
3
4   class Links(scrapy.Spider):
5       name = 'links'
6       start_urls = ['https://www.pagina12.com.ar/secciones/sociedad?page=1']
7
8       def parse(self, response):
9           for link in response.css('h4.title-list'):
10              yield {'url': 'https://www.pagina12.com.ar' + link.css('a').attrib['href']}
11
12          for next_page in response.css('a.next'):
13              yield response.follow(next_page, self.parse)
```

En la línea n°1 se hace la importación de la librería 'scrapy'. Luego en la línea n°4 se hace el llamado a la clase 'Spider' que permitirá llevar a cabo el *scrapping* de las páginas del diario.

En la línea n°5 se establece el nombre del archivo donde se almacenarán las direcciones de las páginas web de cada uno de los artículos. En la línea n°6 se define el *link* en donde se iniciará el proceso de *scrapping*. El URL inicial es el definido entre corchetes luego del código 'start_urls ='. El archivo tendrá una extensión '.json'. Al finalizar cada extracción y antes de empezar con una nueva URL se cambiará el nombre del archivo resultante (el proceso se explica en un apartado posterior).

Entre las líneas n°8 y n°13 se define la función 'parse' que permite extraer todas las páginas web. Para ello se establecieron dos ciclos 'for', uno para recorrer cada una de las páginas y extraer los links y el segundo ciclo para navegar a la siguiente página. En el primer ciclo 'for' se indica el marcador que indica dónde está el *link* correspondiente a cada artículo ('h4.title-list'). Cabe destacar que Página12 tiene los links de los artículos truncados, por lo que se los debe concatenar con el URL de la página principal ('https://www.pagina12.com.ar') con el *link* abreviado que se extrae por medio de 'link.css('a').attrib['href']'. Luego, se continúa revisando la página hasta encontrar el siguiente marcado y repetir el proceso. Cuando se ha extraído la última dirección (no se encuentran más marcadores en la página), se pasa al segundo ciclo 'for' que permite navegar a la siguiente página.

Dado que no se ha establecido un proceso de parada automática, se deberá monitorear la ejecución del archivo para asegurar que sólo se extraigan la cantidad de links deseados.

Archivo settings.py

Este archivo se utiliza para configurar el *scrapper* que permitirá extraer el texto completo de los artículos cuyas direcciones se obtuvieron por medio de 'links.py'. Es un archivo correspondiente a la librería 'scrapy' que permite customizar la ejecución del *scrapper*. En este caso se busca evitar que el comportamiento que se lleve a cabo resulte sospechoso, por lo tanto, se deshabilita la función 'autothrotle_enabled' que limita el tiempo entre solicitudes. Esto también se lleva a cabo como una buena práctica de etiqueta cuando se realiza el proceso de *scrapping*. A sí mismo, también se tuvo en cuenta que se deben respetar todos los 'robot-text' con la finalidad de tener un comportamiento decoroso al extraer la información.

Por último, se estableció que el 'encoding' de la extracción será con 'UTF-8' lo que permitirá extraer el texto de las páginas con los acentos y caracteres latinos reflejados correctamente.

Archivo articles.py

Este archivo se utiliza para extraer la información de cada uno de los archivos y almacenarlo dentro de un único registro en el archivo '.json' que se generará. Los campos que se extraerán para cada artículo son los siguientes:

a. 'section': economy (Economía), society (Sociedad) y world (El Mundo).
b. 'date': fecha del artículo a extraer.
c. 'deck': copete del artículo.
d. 'title': título del artículo.
e. 'lead': resumen del artículo.
f. 'tags': palabras claves del artículo.
g. 'author': autor del artículo.
h. 'article': texto completo del artículo.

Todos los campos mencionados anteriormente se encuentran dentro de la función 'parse' dentro de este archivo. Dicha función, además de extraer los campos citados, lleva a cabo algunas modificaciones antes de almacenar todos ellos en un solo registro del archivo 'society-articles.json'. Las modificaciones que se llevaron a cabo fueron las siguientes:

a. 'date': la fecha se trunca para eliminar el componente horario y sólo se guarda la fecha en formato AAAA-MM-DD.
b. 'author': a todos los artículos que no tengan autor, en este campo se agrega la leyenda "Por: ". Luego, se elimina dicha leyenda para todos los artículos. Esto permite dejar únicamente el nombre del autor o sino un campo vacío para cada uno de los artículos.
c. 'article': se utilizó una función de *list comprehension* para desarrollar toda la función en una sola línea. Se busca el texto de todo el artículo y, a continuación, se eliminan todos los espacios adicionales y saltos de línea (párrafos). Cada uno de los saltos de línea fueron reemplazados por un espacio. Como resultado todo el artículo queda en un solo párrafo.

Archivo Vectorize.ipynb

Este archivo es un *notebook* de Python que contiene el código para unir en un solo archivo y vectorizar los archivos 'economy-articles.json', 'society-articles.json' y 'world-articles.json'.

En el primer bloque de código se importan las librerías que se utilizarán que son 'json', 'joblib' y 'pandas'. También se utilizarán el 'tokenizador' de la librería 'utils.tokenizers', 'CountVectorizer' de 'sklearn.feature_extraction.text' y 'List' de 'typing'.

En el segundo bloque de código se importan los tres archivos '.json' que se mencionan en el primer párrafo de la presente sección a la variable correspondiente. Por tanto, 'economy-articles.json' se asigna a la variable 'economy', 'society-articles.json' a 'society' y 'world-articles.json' a 'world'

En el tercer bloque, se convierten las tres variables a un *dataframes* y luego se los unifica en un único *dataframe* denominado 'df_news'.

En el cuarto bloque, se genera una nueva columna en el *dataframe* denominada 'final' que resulta de la concatenación de las columnas 'deck', 'title', 'lead', 'article' con su correspondiente conversión cadenas de caracteres (*strings*). Para separar los distintos campos se agregan espacios en el medio.

En el quinto bloque, se define la función 'leer_stopwords' la cual arma un listado de *stopwords* a ser detectadas en el texto. A continuación, en el sexto bloque, se hace una limpieza de todas las palabras eliminando las mayúsculas y los acentos, y, además, se arman los bigramas que serán utilizados para analizar el corpus.

En el séptimo bloque, se define el modelo a emplear y luego se aplica a la columna 'final' del *dataframe* donde aparece cada palabra como encabezado de las columnas y la cantidad de veces que aparece cada una de ellas.

Por último, en el octavo bloque se vuelca (*dump*) de las variables generadas por el archivo hacia tres archivos '.joblib' que serán utilizado por el archivo 'Evaluate.ipynb' para la evaluación del modelo generado.

Archivo Evaluate.ipnyb

Este archivo contiene el código para poder evaluar el modelo desarrollado por medio de los artículos citados anteriormente. En el primer bloque de código se cargan todas las librerías de Python que serán requeridas para la ejecución de este.

En el segundo bloque de código se definen las distintas funciones que se aplicarán, a saber:

- **_calcular_auc_por_clase(targets_reales:np.ndarray, targets_preds:np.ndarray) -> Dict[int, float]:** computa la curva ROC y AUC para cada clase. Sus parámetros son:
    - **targets_reales:** Un vector de targets reales representados en *1-hot encoding*.
    - **targets_preds:** Un vector de targets predichos representados en *1-hot encoding*.
    - **Devuelve:** Un diccionario de índice de categoría -> AUC de esa categoría
- **calcular_e_imprimir_auc(clasificador, train_fold_selected, train_targets_binarios_por_clase, test_fold_selected, test_targets_binarios_por_clase):** calcula e imprime el AUC para cada categoría, utilizando el clasificador y los *folds* de entrenamiento y *test*. Sus parámetros son:
    - **clasificador:** un clasificador de la librería 'scikit-learn'.
    - **train_fold_selected:** *fold* de entrenamiento.
    - **train_targets_binarios_por_clase:** categorías del *fold* de entrenamiento, en *1-hot encoding*.
    - **test_fold_selected:** Fold de test
    - **test_targets_binarios_por_clase:** categorías del *fold* de test, en *1-hot encoding*.
- **pesos_de_features(score_fn, train_fold, train_targets_fold):** establece el peso de cada uno de los *features* que componen el *dataset*. Sus parámetros son:
    - **score_fn:** una función que pueda tomar una columna de *feature* y la columna de categoría, y calcular un *score* que mida qué tan bien esa columna predice las categorías. Puede ser cualquier función dentro de *sklearn.feature_selection* como chi2, *mutual_info_classif*, o *relief*.
    - **train_fold:** *fold* de entrenamiento.
    - **train_targets_fold:** un arreglo con el valor de la categoría de cada fila en 'train_target_fold'.

- **imprimir_features_con_pesos(score_fn, train_fold, train_targets_fold, nombres_features, top_n=-1):** evalúa cada columna del *dataset* con el fin de determinar su utilidad en la clasificación del *dataset*. Sus parámetros son:
    - **score_fn:** función que pueda tomar una columna de *feature* y la columna de categoría, y calcular un score que mida que tan bien esa columna predice las categorías. Puede ser cualquier función dentro de *sklearn.feature_selection* como chi2, *mutual_info_classif*, o *relief*.
    - **train_target_fold:** matriz con columnas a evaluar, excluyendo la columna de categoría de cada fila.
    - **train_targets_fold:** arreglo con el valor categoría de cada fila en 'train_target_fold'.
    - **nombre_features:** los nombres de cada columna en 'train_target_fold'.
    - **top_n:** cuántos de los mejores scores imprimir. -1 imprime todos.
- **nombres_features_seleccionadas(selector_features, nombres_features):** retorna los nombres de las columnas seleccionadas como mejores por selector_features. Sus parámetros son:
    - **selector_features:** función de 'sklearn' que puede evaluar los scores de columna y seleccionar las mejores. Puede ser *SelectKBest*, *GenericUnivariateSelect* o *SelectPercentile*.
    - **nombres_features:** lista de nombres de columnas. El orden de las columnas tiene que ser el mismo que el de la matriz con la que se evaluó a 'selector_features'.
    - **Devuelve:**
        - **new_features:** lista de nombres de *features* que se corresponde con las seleccionadas por 'selector_features'.

En el tercer bloque de código se leen los *datasets* resultantes del archivo 'Vectorize.ipynb' y, además, se configura el clasificador que se utilizará. Se optó por correr dos validaciones distintas para comparar los resultados. Se emplearán los métodos de *5-fold cross-validation* y *gradient-boosting* para evaluar los resultados del modelo.

Por último, el cuarto bloque de código es el que lleva a cabo la evaluación del modelo.

Descripción del Proceso

Como se observa en la sección 'Archivo.py', para poder llevar a cabo el ejercicio de extracción de los artículos de las tres secciones del diario Página12, se desarrolló un *web crawler* desde cero teniendo en cuenta la estructura de las páginas. Los pasos llevados a cabo fueron los siguientes:

1. En una primera instancia se llevó a cabo la extracción de todas las URL correspondientes a aproximadamente un año de noticias de cada una de las secciones. Esto corresponde a trescientas páginas y en total se extraerán tres mil trescientos (3.300) *links*. Se decide iniciar la ejecución del proceso de extracción en la sección "Economía", por tanto, la URL de inicio será: https://www.pagina12.com.ar/secciones/economia?page=1. Las direcciones de los artículos se extraerán y almacenarán en un archivo denominado 'links.json'.
2. Luego de detener la ejecución del archivo 'links.py' cuando se termine la extracción de los *links* de las trescientas páginas se cambiará el nombre del archivo resultante a "economy-links.json". El archivo resultante contendrá un listado de *links* como el que se muestra a continuación:

```
[
    {
        "url": "https://www.pagina12.com.ar/365361-los-agro-dolares-alcanzan-el-record-del-siglo"
    },
    {
        "url": "https://www.pagina12.com.ar/365364-acuerdo-con-cuba-para-transferir-tecnologia"
    },
    {
        "url": "https://www.pagina12.com.ar/365365-recuperacion-con-igualdad-de-genero"
    },
    {
        "url": "https://www.pagina12.com.ar/365419-recuperacion-industrial-un-mensaje-de-campana"
    },
```


3. Repetir los pasos n°1 y n°2 para las dos secciones restantes (Sociedad, El Mundo) tomando como páginas de inicio las siguientes:
    a. https://www.pagina12.com.ar/secciones/el-mundo?page=1
    b. https://www.pagina12.com.ar/secciones/sociedad?page=1
4. Los archivos resultantes de la ejecución del paso n°3 deberán renombrarse como 'society-links.json' y 'world-links.json' respectivamente.
5. Luego de obtener los tres archivos con los *links* correspondientes a tres mil trescientos artículos por sección, se debe seleccionar uno de ellos y ejecutar el archivo 'articles.py' donde se coloca la ubicación del archivo correspondiente a la sección "Economía" denominado 'economy-links.json' en la línea n°9 del código.
6. A continuación, se ejecuta el archivo que entregará como archivo de salida uno denominado 'articles.json'. Dicho archivo se deberá renombrar a 'economy-articles.json' antes de repetir la ejecución para las siguientes secciones. El archivo resultante contendrá un listado con el contenido de los artículos como se observa a continuación:


```
[
    {
        "section": "economy",
        "date": "2021-09-02",
        "deck": "Por exportaciones, ingresaron 3000 millones en agosto ",
        "title": "Los agro-dólares alcanzan el récord del siglo",
        "lead": "La cámara CIARA CEC informó además que en los primeros ocho meses del año se alcanzaron "máximos históricos" para el sector. ",
        "tags": [],
        "author": "",
        "article": "El boom de precios de los commodities agropecuarios sigue dándole a Argentina un nivel inédito de ingresos de dólares por exportaciones. Un informe de la Cámara de la Industria Aceitera de la República Argentina (Ciara) y el Centro de Exportadores de Cereales (CEC) , precisó que las ventas externas de cereales, oleaginosas y sus derivados alcanzaron tanto en agosto último como en el acumulado de los primeros ocho meses del año registros máximos históricos para el sector. En ese sentido, las empresas del sector agroexportador liquidaron el mes pasado 3.049,78 millones de dólares, que significaron una marca histórica para el mes. Así las cosas, los dólares del agro acumulados en los primeros ocho meses alcanzaron los 23.229.238.627 de dólares. \"El monto de agosto pasado resulta récord para ese mes en las estadísticas desde comienzos de este siglo y en toda la serie histórica\", precisaron en un comunicado Ciara y CEC, entidades que representan el 48 por ciento de las exportaciones argentinas. Lo liquidado en agosto significó un 74,97 por ciento superior a lo registrado un año atrás, pero fue un 13,35 por ciento menor a lo ingresado en julio último. Asimismo, el ingreso de divisas de los primeros ocho meses del año representó un crecimiento del 74 por ciento respecto del mismo período de 2020, y récord absoluto para el mismo período desde comienzos de este siglo. \"Esos resultados se alcanzaron por los significativos esfuerzos logísticos a los que obligó la histórica baja del río Paraná . No obstante, los precios internacionales de los commodities resultaron un atractivo que le dieron fluidez a las ventas de los productores\", precisaron desde la entidad. Naturalmente, las liquidaciones del sector están apuntalando las Reservas del Banco Central, lo que le da aire al Gobierno para sostener el manejo de la política cambiaria con respaldo monetario. Pero este boom de agro tiene una particularidad, no sólo es un fenómeno de alza de precios de la soja, sino que alcanza a todos los granos que cotizan en el mercado de Chicago. El complejo oleaginoso-cerealero, incluyendo al biodiésel y sus derivados, aportó el año pasado el 48 por ciento del total de las exportaciones de la Argentina, según datos del Instituto Nacional de Estadística y Censos (Indec). En este contexto, el principal producto de exportación del país es la harina de soja (14,2 por ciento del total), que es un subproducto industrializado generado por este complejo agroindustrial, que tiene actualmente una elevada capacidad ociosa cercana al 50 por ciento. El segundo producto más exportado el año pasado, de acuerdo con el Indec, fue el maíz (11 por ciento) y el tercero fue el aceite de soja (6,9 por ciento). Qué es la liquidación de divisas? Una vez que los productores cosechan en el grano en el campo, se lo venden a las firmas exportadoras, que cargan el producto en barcos y lo venden al exterior, ya sea como granos o como producto procesado. La mayor parte del ingreso de divisas en este sector se produce con bastante antelación a la exportación, anticipación que ronda los 30 días en el caso de la exportación de granos y alcanza hasta los 90 en el de la de aceites y harinas proteicas. "
    },
    {
        "section": "economy",
        "date": "2021-09-02",
        "deck": "Se busca que la isla desarrolle su sector agropecuario",
        "title": "Acuerdo con Cuba para transferir tecnología",
        "lead": "Los gobiernos avanzan en un plan en el que Argentina aporta tecnología y maquinarias y Cuba aumenta las compras de otros productos.",
        "tags": [
            "Cuba",
            "Jorge Neme",
            "acuerdo comercial"
        ],
        "author": "Natalí Risso",
        "article": "El gobierno avanza en un acuerdo comercial de transferencia de tecnología agroalimentaria, know how y maquinaria con Cuba , a fin de que la isla desarrolle su sector agropecuario a través de tecnología argentina. Participarán como proveedores el INTA, YPF Agro, productores y empresas argentinas con potencial exportador. Si bien aún no finalizó la etapa de negociación, el proyecto podría incluir
```


7. Repetir los pasos n°5 y n°6 para las secciones restantes, utilizando los archivos 'society-links.json' y 'world-links.json' para la ejecución y renombrando los archivos de salida como 'society-articles.json' y 'world-articles.json' respectivamente.

Luego de extraídos el contenido de los artículos para las tres secciones, se debe dar inicio al procesamiento de estos archivos para poder armar el clasificador.

8. Ejecutar el archivo 'Vectorize.ipynb', importando los tres archivos que contienen la información extraída de cada sección del diario (economy-articles.json, society-articles.json y world-articles.json). El resultado de este paso se utilizará para llevar a cabo la evaluación del modelo. Se habrán generado los archivos 'vectores.joblib', 'targets.joblib' y 'features.joblib'.
9. Ejecutar el archivo 'Evaluate.ipynb' para el proceso de *5-fold cross-validation*. Para ello se debe establecer la cantidad de *folds* a cinco. También se debe definir el 'MAX_FEATURES'. En este caso se optó por utilizar dos mil (2.000) 'MAX_FEATURES'.
10. Registrar los resultados obtenidos.
11. Repetir los pasos n°9 y n°10 para la ejecución del proceso de *gradient boosting*, estableciendo la cantidad de estimadores a quinientos.

Casos No Vistos

Al finalizar la ejecución del archivo 'Evaluate.ipynb', se debe llevar a cabo el análisis del código para casos no utilizados para el entrenamiento. Para ello, se decidió utilizar como conjunto de entrenamiento a los artículos correspondientes al período 13 de septiembre 2020 hasta el 31 de julio de 2021 y como conjunto de prueba los artículos comprendidos entre el 1 de agosto de 2021 y el día 2 de septiembre de 2021.

Con los nuevos conjuntos de entrenamiento y prueba definidos, se modificó levemente el código del archivo 'Vectorize.ipynb' y se generó un nuevo archivo denominado 'Evaluate Corte Agosto.ipynb' para ajustarlo a las nuevas condiciones. Dado que se estará utilizando un conjunto de prueba con las fechas pre-establecidas y se utilizará para el entrenamiento los artículos anteriores al 1 de agosto de 2021.

Para la evaluación del nuevo conjunto de prueba se llevaron a cabo los siguientes pasos utilizando el archivo 'Evaluate Corte Agosto.ipynb':

1. Cargar las librerías de funciones que serán utilizadas (análogas a las del archivo 'Evaluate-ipynb').
2. Definir las funciones a utilizar (análogas a las del archivo 'Evaluate-ipynb').
3. Leer los *datasets* 'vectores.joblib', 'targets.joblib' y 'features.joblib' y asignarlos a las variables 'vectores', 'nombres_targets' y 'nombres_features' (paso análogo al del archivo 'Evaluate-ipynb').
4. Convertir las categorías a números, esto es modificar los nombres de las secciones 'economy', 'society' y 'world' a números enteros, empezando por el 0. Transformar los targets en N columnas, una por cada categoría, donde la categoría correcta tiene como valor '1' y todas las demás columnas en esa fila tienen '0'. Dado que AUC se calcula sobre dos categorías, se utilizará esto para calcular un AUC por cada una de ellas.
5. Realizar la separación entre el conjunto de entrenamiento y el de pruebas conforme a la regla establecida anteriormente. Esto resulta en ochomil seiscientos ochenta y un registros para el entrenamiento y mil doscientos sesenta y nueve registros de prueba.
6. Entrenar el modelo utilizando un modelo de *gradient boosting*, utilizando quinientos estimadores y estableciendo un 'random_state' igual a '1234' para que los resultados siempre sean los mismos a la hora de ejecutar el código. Además, se estableció el número máximo de *features* en dos mil, al igual que se había establecido para el caso de *5-fold cross-validation*.
7. Realizar el cálculo de la precisión del modelo (*accuracy*).
8. Calcular el AUC para cada una de las categorías y, a continuación, realizar la matriz de confusión.

Análisis de los Resultados

5-fold cross-validation

Como resultado del *5-fold cross-validation* los resultados obtenidos para cada uno de los *folds* fueron los que se muestran en la tabla a continuación. Cabe recordar que se utilizan el 80% de los datos para entrenar el modelo y el 20% para evaluarlo en cada uno de los *folds.*

	AUC			Accuracy
	Economía	Sociedad	El Mundo	
Fold 1	0.9606	0.9163	0.9330	0.9313
Fold 2	0.9663	0.9136	0.9341	0.9348
Fold 3	0.9614	0.9326	0.9337	0.9369
Fold 4	0.9678	0.9242	0.9280	0.9338
Fold 5	0.9667	0.9216	0.9201	0.9333
Promedio	0.9646	0.9217	0.9298	0.9340

La matriz de confusión resultante para cada *fold* fue la siguiente

- Fold 1:

	Economía	Sociedad	El Mundo
Economía	628	30	2
Sociedad	17	611	32
El Mundo	8	47	605

- Fold 2:

	Economía	Sociedad	El Mundo
Economía	629	26	5
Sociedad	14	605	41
El Mundo	4	39	617

- Fold 3:

	Economía	Sociedad	El Mundo
Economía	631	25	4
Sociedad	14	619	27
El Mundo	5	50	605

- Fold 4:

	Economía	Sociedad	El Mundo
Economía	638	17	5
Sociedad	16	615	29
El Mundo	8	56	596

- Fold 5:

	Economía	Sociedad	El Mundo
Economía	639	18	3
Sociedad	16	619	25

El Mundo	10	60	590

- Promedio:

	Economía	Sociedad	El Mundo
Economía	633	23	4
Sociedad	15	614	31
El Mundo	7	50	603

Como resultado del análisis con el modelo de *5-fold cross-validation* se obtuvieron resultados sumamente satisfactorios dado que el *accuracy* promedio fue del 0.9340. También se detectó una leve tendencia donde el modelo detectaba en forma correcta los artículos correspondientes a la sección "Economía" por encima de "Sociedad" y "El Mundo". Sin embargo, se puede considerar que los resultados en todos los casos han sido similares.

$$\begin{bmatrix} 633 & 15 & 4 \\ 23 & 614 & 31 \\ 7 & 50 & 603 \end{bmatrix}$$

Al analizar el promedio de los cinco *folds,* utilizando la matriz de confusión, se llega a las siguientes conclusiones respecto de los casos de prueba:

- 633 artículos correspondientes a la sección "Economía" fueron clasificados correctamente.
- 614 artículos correspondientes a la sección "Sociedad" fueron clasificados correctamente.
- 603 artículos correspondientes a la sección "El Mundo" fueron clasificados correctamente.
- 15 artículos correspondientes a la sección "Economía" fueron clasificados como pertenecientes a la sección "Sociedad".
- 7 artículos correspondientes a la sección "Economía" fueron clasificados como pertenecientes a la sección "El Mundo".
- 23 artículos correspondientes a la sección "Sociedad" fueron clasificados como pertenecientes a la sección "Economía".
- 50 artículos correspondientes a la sección "Sociedad" fueron clasificados como pertenecientes a la sección "El Mundo".
- 4 artículos correspondientes a la sección "El Mundo" fueron clasificados como pertenecientes a la sección "Economía".
- 31 artículos correspondientes a la sección "El Mundo" fueron clasificados como pertenecientes a la sección "Sociedad".

Casos No Vistos

Para poder determinar el verdadero poder de predicción del modelo desarrollado, se utilizó la división citada en la sección 'Casos No Vistos' donde se tomaron los artículos posteriores al 1 de agosto de 2021 para la prueba y el resto para el entrenamiento del modelo. El resultado de la prueba fue el siguiente:

AUC			Accuracy
Economía	Sociedad	El Mundo	
0.9536	0.8929	0.9311	0.9114

$$\begin{bmatrix} 312 & 16 & 0 \\ 21 & 558 & 50 \\ 1 & 20 & 241 \end{bmatrix}$$

Al analizar el promedio de los cinco *folds,* utilizando la matriz de confusión, se llega a las siguientes conclusiones respecto de los casos de prueba:

- 312 artículos correspondientes a la sección "Economía" fueron clasificados correctamente.
- 558 artículos correspondientes a la sección "Sociedad" fueron clasificados correctamente.
- 241 artículos correspondientes a la sección "El Mundo" fueron clasificados correctamente.
- 21 artículos correspondientes a la sección "Economía" fueron clasificados como pertenecientes a la sección "Sociedad".
- 1 artículos correspondientes a la sección "Economía" fueron clasificados como pertenecientes a la sección "El Mundo".
- 16 artículos correspondientes a la sección "Sociedad" fueron clasificados como pertenecientes a la sección "Economía".
- 20 artículos correspondientes a la sección "Sociedad" fueron clasificados como pertenecientes a la sección "El Mundo".
- 0 artículos correspondientes a la sección "El Mundo" fueron clasificados como pertenecientes a la sección "Economía".
- 50 artículos correspondientes a la sección "El Mundo" fueron clasificados como pertenecientes a la sección "Sociedad".

Conclusiones

Luego de realizado el análisis de todo el corpus de artículos utilizando dos técnicas diferentes (*5-fold cross validation* y *gradient boosting*) se pudo observar que los resultados de *accuracy* obtenidos en ambas pruebas son similares, siendo el de *5-fold cross validation* levemente mejor que el de *gradient boosting*. Este resultado era esperable ya que en el caso de *5-fold cross validation* se utilizó todo el corpus de artículos para su entrenamiento y evaluación, mientras que en el de *gradient boosting* se entrenó el modelo con algunos casos y luego se hizo la prueba con casos no vistos.

Al comparar los resultados de ambos modelos, se obtiene los siguientes resultados:

	AUC			Accuracy
	Economía	Sociedad	El Mundo	
Promedio-CV	0.9646	0.9217	0.9298	0.9340
Gradient Boosting	0.9536	0.8929	0.9311	0.9114

Como se puede observar, y fue mencionado anteriormente, los resultados obtenidos por *5-fold cross validation* son superiores a los del modelo *gradient boosting* para las secciones 'Economía' y 'Sociedad' mientras que para la sección 'El Mundo' el modelo de *gradient boosting* arroja un resultado levemente mejor. Cabe destacar que dentro del período bajo análisis en el caso de *gradient boosting* ocurrieron las elecciones denominadas PASO, donde los resultados ocasionaron una gran cantidad de artículos dentro de la sección 'Sociedad' con información que no había sido parte del conjunto de entrenamiento. Este pudiera ser el factor determinante de la gran variación en el resultado entre los dos modelos para dicha sección. La diferencia detectada para la sección 'El Mundo' es sumamente pequeña y, por tanto, se puede

llegar a considerar que los resultados obtenidos son iguales. En el caso de 'Economía' la precisión del modelo de *5-fold cross validation* es levemente superior a la del modelo de *gradient boosting* pero dicha diferencia también puede ser explicada por los eventos ocurridos durante el período utilizado como prueba por efecto de las elecciones que acontecieron.

Tomando en consideración lo expuesto anteriormente, se concluye que ambos métodos son buenos selectores de la sección a la que corresponde un artículo basándose en una máxima cantidad de *features* establecida en dos mil. Además, también se pudo observar qué si ocurren eventos de alto impacto, el modelo empieza a perder su precisión. Esto resulta análogo al caso de series temporales donde el poder predictor de los modelos desciende con el crecimiento de la distancia entre el conjunto de entrenamiento y el de prueba. En el caso del análisis de noticias periodísticas, los factores exógenos y coyunturales son los que afectan el modelo y, el corpus de entrenamiento tiene poco poder predictor para dichos casos.