“Machine Learning” con Python

Breve descripción:

Este componente formativo está enfocado en reconocer todos los elementos necesarios para aplicar el modelo “Machine Learning” a través de un ejemplo enfocado en la selección y utilización de los diferentes algoritmos de clasificación que se encuentran dentro de la categoría del aprendizaje supervisado.

Noviembre 2023

Tabla de contenido

Introducción

Le damos la bienvenida a este componente formativo denominado “Machine Learning” con Python, para comenzar el recorrido por el mismo, conozca en qué consiste el “Machine Learning” y por qué la pertinencia de Python.

Video 1. “Machine Learning” con Python

Enlace de reproducción del video

Síntesis del video: “Machine Learning” con Python
¿Sabías que a través de los años se ha ido aumentando el interés y la utilización del “Machine Learning”? Así es, su implementación cada vez es más constante; puede decirse que esta se ha posicionado como una de las disciplinas que se están

aplicando casi en todas las áreas de investigación académica e industrial.

"Machine Learning" hace referencia a un grupo de algoritmos que permiten realizar la identificación de patrones presentados en colección de datos, a partir de los cuales se pueden generar diversas estructuras y modelos que permiten la predicción de información que no había podido ser analizada. Es por ello que se han generado una serie de herramientas mediante las cuales se pueden acceder a métodos predictivos muy poderosos. Un ejemplo claro de ello es el lenguaje de programación Python.

Finalmente, se escucha hablar de términos como "data mining", inteligencia artificial, "data science" o "Machine Learning", los cuales, de cierta manera, se refieren a lo mismo. Sin embargo, se debe decir que las técnicas que se utilizan en "Machine Learning" hacen parte de una de las muchas formas que se necesitan para mezclar, extraer, entender y darle valor a los datos.

El objetivo de este componente es plantear un ejemplo parecido a lo que un analista de datos enfrentaría, partiendo de una colección de datos con cierta preparación. Es importante recordar que la preparación de los datos es uno de los procesos mediante el cual se requiere mayor cantidad de tiempo y es una etapa previa a la utilización de técnicas de "Machine Learning".

Con el desarrollo del presente, entonces, se busca crear un modelo para predecir con cierta efectividad el comportamiento que tomarían nuevas entradas, para esto se debe tener claridad frente al problema, explorar los datos, establecer métricas para evaluar los resultados y a partir de allí establecer estrategias para ajustar y mejorar el

modelo si es necesario, en este orden de ideas se suministrarán las bases para aplicar dicho modelo en la solución de problemas reales del contexto.

1. Implementar entorno de trabajo con Python

Python es un lenguaje de programación altamente utilizado y líder dominante en todo lo relacionado con la estadística, la "data mining" y por supuesto el "Machine Learning", es un "software" de libre uso, lo que permite que un sinnúmero de personas puedan hacer las implementaciones de sus algoritmos, obteniendo de esta manera una gran cantidad de librerías utilizadas en la implementación de casi todas las técnicas de "Machine Learning" que existen actualmente.

El entorno de trabajo utilizado es Anaconda, el cual está basado en el lenguaje de programación Python. Este programa se puede descargar gratuitamente desde su página oficial y al descargarlo se visualiza una serie de aplicaciones para el manejo de proyectos de ciencia de datos.

Para este caso se usará la aplicación JupiterLab, incorporada en Anaconda, que es ampliamente usada para la creación de los denominados "Notebook". Con esta herramienta se realizará toda la programación basada en el lenguaje de programación Python.

Para realizar la instalación de Anaconda se debe observar el video propuesto:

Video 2. Instalación de Anaconda

Enlace de reproducción del video

Síntesis del video: Instalación de Anaconda
Videotutorial en el que se explica cómo descargar el programa Anaconda, teniendo en cuenta el sistema operativo. También, expone el paso a paso del proceso de instalación y finaliza mostrando cómo debe aparecer el programa cuando se ha hecho correctamente el procedimiento.

Ahora bien, para desplegar un entorno de trabajo con JupiterLab se deben seguir los siguientes pasos dados en el recurso:

Video 3. Pasos para un entorno de trabajo con JupiterLab

Enlace de reproducción del video

Síntesis del video: Pasos para un entorno de trabajo con JupiterLab
Videotutorial en el que se explica cómo poner en marcha el entorno de trabajo con el aplicativo JupiterLab dentro de Anaconda Navigator, dando clic en el botón "Launch". Una vez se ha presionado, se abre un sitio web con un entorno en el que se explica cómo se navega, trabaja y organiza un proyecto, el cual inicia con la creación de un directorio o ruta para los archivos que se van a crear. Luego, dentro de esa ruta, se selecciona "Notebook", el cual es el aplicativo en el que se trabaja el proyecto, mostrando cómo renombrar estos archivos y cómo funcionan a través de celdas, ya sean de tipo código (interpretado por Python) o de

tipo comentario (utilizados para documentar el proyecto). Finalmente, se explica el panel de herramientas como cierre del videotutorial.

Después de realizar las instalaciones de las herramientas para el desarrollo del proyecto de “Machine Learning” se requiere del uso de ciertas librerías especializadas para el tratamiento de los datos, de las cuales se hablará a continuación.

Librerías para trabajar “Machine Learning”

La alta demanda del lenguaje de programación Python ha desbordado toda serie de librerías que ayudan a múltiples tareas, facilitando el trabajo y haciéndolo cada vez más rápido y eficiente. Por tanto y para este caso, se utilizarán librerías propias de proyectos de ciencia de datos. En el siguiente video se expone el proceso de importación:

Video 4. Importación de librerías

Enlace de reproducción del video

Síntesis del video: Importación de librerías
Videotutorial que busca presentar, a través de un ejemplo, la importación de librerías y cómo utilizarlas. Para ello se creó un archivo con varias celdas. En la primera, se muestra cómo se importan las librerías mediante la palabra reservado "import", el nombre de la librería y la palabra as (para renombrarla con un nombre corto, aunque algunas de ellas ya están predeterminadas). Luego, se presenta cómo ejecutar dicha celda y comprobar que la importación quedó correcta. En la segunda celda, se muestra cómo transformar datos (que son del tipo clave valor) en una estructura de filas y columnas. Para ello se utiliza la librería Pandas, escribiéndola cómo se renombró en la primera celda y se visualiza con la palabra "print". En la tercera celda, se utilizará la librería Matplotlib para analizar unas líneas de código y variables para que arroje un gráfico comparativo. Finalmente, el proceso del videotutorial se resume en la importación de las librerías, cómo se renombran y cómo se utilizan en las celdas donde se deben aplicar.

Entre los tipos de librerías se encuentran:

- **Pandas:** es una librería de Python especializada en el manejo y análisis de estructuras de datos; dentro de sus principales usos se puede encontrar:
    - Define nuevas estructuras de datos basadas en los "arrays" de la librería NumPy pero con nuevas funcionalidades.
    - Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.

- Permite acceder a los datos mediante índices o nombres para filas y columnas.
- Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
- Permite trabajar con series temporales.
- Realiza todas estas operaciones de manera muy eficiente.

Para realizar la importación de la librería Pandas y empezar a trabajar con sus funcionalidades, se debe utilizar el siguiente comando: **import pandas as pd**

A continuación, se observa la potencia de esta librería mediante la cual con una sola línea de código, realiza el cargue de una colección de datos en formato CSV; el comando requerido es el siguiente: **df=pd.read_csv('spam.csv')**

Para la visualización de los datos se utiliza el siguiente: **df.head()**

El resultado obtenido es:

"Category"	"Message"
0 ham	"Go until jurong point, crazy.. Available only ..."
1 ham	"Ok lar... Joking wif u oni..."
2 spam	"Free entry in 2 a wkly comp to win FA Cup fina..."
3 ham	"U dun say so early hor... U c already then say..."
4 ham	"Nah I don't think he goes to usf, he lives aro..."

Este comando permitirá visualizar una muestra de los datos; la librería Pandas es muy útil para todo el manejo y manipulación de datos, ya que con muy pocas líneas de código se puede realizar procedimientos complejos.

- **Numpy:** es una librería muy importante cuando se trabaja con proyectos de ciencia de datos en Python. Se especializa en los cálculos numéricos y analítica de datos, específicamente cuando se trata de grandes volúmenes de información.

  Esta librería trae incorporada una nueva clase de objetos que se denominan "arrays" que se utilizan para representar colecciones de datos que son de un mismo tipo en varias dimensiones y funciones que son bastante eficientes para realizar su manipulación.

La librería Numpy es hasta 50 veces más rápida comparada con las listas que vienen ya predeterminadas con el lenguaje Python. Esto hace que sean muy buena opción cuando se trata del procesamiento de vectores y matrices que son de gran dimensión.

Para trabajar con esta librería se debe importarla de la siguiente manera:

Import numpy as np

robustas.

Normalmente una lista de Python se define de la siguiente manera:

listaPython=[2,4,6,8,10]

Para realizar la conversión a un "array", se debe envolver dentro del método array() de la siguiente manera.

np.array(listaPython)

El resultado que se obtiene es el siguiente.

array([2, 4, 6, 8, 10)]

Los dos tipos de arreglos que se pueden encontrar en NumPy son vectores y matrices; en el anterior ejemplo, se veía la conversión de una lista normal de Python a un vector en NumPy y un ejemplo de matrices se puede ver a continuación:

matriz = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]

np.array(matriz)

El resultado es una matriz de una dimensión determinada por tres filas y tres columnas y se ve de la siguiente manera.

array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

Por ejemplo, si se quisiera realizar un matriz de dimensión 5 por 5 con números aleatorios simplemente se ejecuta el siguiente comando:

matriz_aleatoria=np.random.rand(5,5)

Para visualizar el resultado simplemente se debe utilizar el comando print de la siguiente manera.

print(matriz_aleatoria)

El resultado obtenido sería el siguiente:

[[0.29851396 0.65165827 0.88150781 0.15657903 0.37648232]

[0.33269701 0.4549997 0.68407767 0.68343099 0.77983704]

[0.74589421 0.53109264 0.83403166 0.56529856 0.40442153]

[0.42269337 0.46910944 0.95015784 0.25355503 0.13611851]

[0.5528267 0.87792271 0.84945654 0.26660765 0.68067849]]

Con esta robusta librería se pueden realizar todo tipo de operaciones basadas en arreglos unidimensionales y multidimensionales.

- **Matplotlib**: es una biblioteca muy útil para realizar gráficos estáticos, animados e interactivos. Los gráficos son muy útiles para hacer análisis de información, permitiendo agrupar grandes volúmenes de datos en representaciones visuales que permiten una mejor comprensión.
  Para usar esta librería se debe realizar la importación de la siguiente forma:
  import matplotlib.pyplot as plt

A continuación, se ve un ejemplo de un gráfico que realiza la simulación de ventas de diferentes productos, mostrando la participación porcentual de cada uno de ellos, y para este caso se usará un gráfico muy útil para visualizar la participación porcentual de cada uno de ellos con respecto al total; estos son los conocidos como gráficos de torta.

Para realizar este ejemplo es necesario realizar los siguientes pasos:

a. **Primero**. Realice la importación de las librerías en Python de la siguiente manera.
   import matplotlib.pyplot as plt.

b. **Segundo.** Defina las etiquetas que representan las salidas de los datos, para este ejemplo serán productos que representan frutas.

etiquetas = 'Manzanas', 'Peras', 'Mangos', 'Fresas'

c. Tercero. Establezca los valores que representan cada uno de los productos definidos.

valores = [25, 15, 45, 5]

d. **Cuarto**. Si quiere resaltar una rebanada de la torta, por ejemplo las peras, se debe establecer una variable de la siguiente manera.

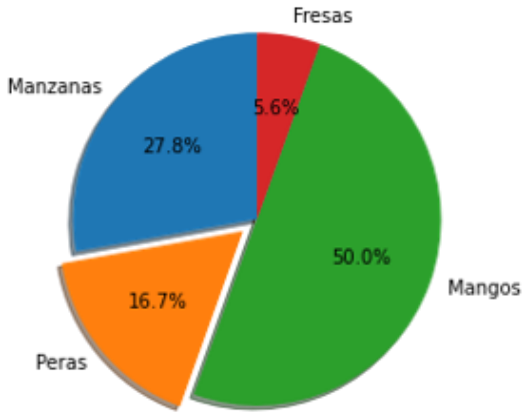
rebanada=(0,0.1,0,0)

e. **Quinto**. Como se observa en el paso anterior, el 0.1 establece una separación del producto número dos correspondiente a las peras, de acuerdo con el número al aumentar el valor la separación de la torta será mayor.

f. **Sexto.** Para visualizar el gráfico use las siguientes líneas de código. Ubicando cada una de las variables definidas anteriormente como se muestra a continuación.

fig1, ax1 = plt.subplots() ax1.pie(valores, explode=rebanada, labels=etiquetas, autopct='%1.1f%%', shadow=True, startangle=90) ax1.axis('equal')

g. **Séptimo**. Finalmente, el resultado que se obtiene es el siguiente:



Vemos que la visualización de los datos se puede realizar muy fácilmente, de igual manera se pueden realizar múltiples gráficos como histogramas de frecuencia, barras, diagrama de caja y bigotes, dispersión y, en general, muchos tipos de gráficos, que de acuerdo con el problema, se pueden implementar.

- **Scikit-learn**: esta es una librería muy importante para todo lo que tiene que ver con aprendizaje automático. Esta biblioteca ofrece una estructura muy fácil y a la vez muy robusta que permite la creación de modelos de "Machine Learning". Se utiliza en el aprendizaje, transformación y predicción desde una colección de datos específica y se pueden utilizar igualmente en el diseño de modelos de agrupación, regresión y clasificación; por otro lado, también es utilizada para el procesamiento y análisis de tipo estadístico, evaluando los modelos entre muchas más funcionalidades.

  Se realizará a continuación un pequeño ejemplo donde se implemente esta librería, dependiendo la siguiente colección de datos, se requiere realizar el aprendizaje de los datos, crear el modelo y finalmente realizar una predicción de nuevas entradas.

La colección de datos con la que se realizará el ejemplo es la siguiente:

Tabla 1. Colección de datos de ejemplo

Ítem	Salario (millones)	Hijos (unidades)	Estado	Clase
1	5	2	Casado(1)	Aprobado(1)
2	4	1	Casado(1)	Aprobado(1)
3	3.5	0	Soltero(2)	Aprobado(1)
4	1.5	4	Casado(1)	Reprobado(0)
5	1	3	Soltero(2)	Reprobado(0)
6	0.8	2	Casado(1)	Reprobado(0)

Esta es una pequeña muestra que simula si se aprueba o reprueba un crédito de una entidad bancaria, en la cual se tienen 6 ejemplos o instancias con tres características o atributos, denominados salario, hijos y estado, y estos se clasifican en dos clases aprobado y reprobado.

Se quiere realizar un modelo para realizar la predicción dependiendo a nuevas entradas, si se aprueban o reprueban créditos. Para realizar este ejemplo es necesario realizar los siguientes pasos:

a. **Paso 1.** Tabular los datos correspondientes a los atributos, de la siguiente manera:

   creditos = [[5, 2, 1],[4, 1, 1],[3.5, 0, 2] [1.5, 4, 1],[1, 3, 2],[0.8, 2, 1]]

b. **Paso 2.** Convertir la lista en un "array" multidimensional para ello realice la importación de la librería NumPy así.

   import numpy as np

Luego se ejecuta este comando para convertir la lista en una matriz.

np.array(creditos)

El resultado es una matriz correspondiente a tres columnas y seis filas.

array([[5. , 2. , 1.],
[4. , 1. , 1.]
[3.5, 0. , 2.],
[1.5, 4. , 1.],
[1. , 3. , 2.],
[0.8, 2. , 1.]])

Almacene el resultado en una nueva variable:

instancias=np.array(creditos)

Cómo se puede observar se están almacenando solo los atributos, y en una variable independiente se almacenan los resultados o clases.

c. **Paso 3**. Realice la tabulación de la clase, en una variable separada y coloque la tabulación de las clases de la siguiente manera.

etiquetas=np.array([1,1,1,0,0,0])

d. **Paso 4**. Importar las librerías necesarias para seleccionar el algoritmo de clasificación que se utilizará en la creación del modelo, que para nuestro ejemplo será el de árbol de decisión.

from sklearn import tree

e. **Paso 5.** Seleccione el algoritmo de árbol de decisión de la siguiente manera, en la variable clasificador se armará el modelo.

clasificador=tree.DecisionTreeClassifier()

f. **Paso 6**. Entrenamiento. Para realizar el aprendizaje de la colección de datos use el comando **fit**, de la siguiente manera, se le indican las instancias o entradas y las etiquetas o salidas:

clasificador.fit(instancias, etiquetas)

g. **Paso 7.** Predicción. Una vez establecido nuestro modelo proceda a realizar las predicciones con el comando **predict** con valores existentes o de prueba y valores nuevos para ver los resultados.

nuevaEntrada=[[5, 2, 1]]

res=clasificador.predict(nuevaEntrada)

if res==1: print('Aprobado')

else: pritn('Reprobado')

El resultado para la siguiente entrada es Aprobado (1), esta entrada nueva corresponde a información con la que el modelo fue entrenado, lo ideal es dejar un porcentaje de los datos para pruebas con los que no se entrene el modelo para verificar la eficiencia, normalmente se deja un veinte por ciento para esta labor.

h. **Paso 8**. Finalmente, quedo todo el diseño de un modelo usando el algoritmo clasificatorio de árboles de decisión.

#Importar las Librerias

import numpy as np

from sklearn import tree

 # Definir las Instancias

creditos = [[5, 2, 1],[4, 1, 1],[3.5, 0, 2],[1.5, 4, 1],[1, 3, 2],[0.8, 2, 1]

instancias=np.array(creditos)

 # Definir las Salidas etiquetas=np.array([1,1,1,0,0,0])

 # Seleccionar el algoritmo de Clasificacion

clasificador=tree.DecisionTreeClassifier()

 # Realizar el Entrenamiento

clasificador.fit(instancias, etiquetas)

 # Realizar las Predicciones

nuevaEntrada=[[5, 2, 1]]

res=clasificador.predict(nuevaEntrada)

if res==1:

print('Aprobado')

else:

pritn('Reprobado')

Resultado: Aprobado

- **NKLT**: el Natural Language Toolkit (NLTK) es un conjunto de librerías y programas para Python que permiten llevar a cabo muchas tareas relacionadas con el Procesamiento del Lenguaje Natural. Muchas de las tareas que se necesitarán hacer ya estarán programadas de manera eficiente en NLTK y se podrán usar directamente en los programas. Además de los programas, se distribuyen también corpus y otros datos lingüísticos. Es una plataforma muy útil tanto para la enseñanza como para el desarrollo y la investigación.

Las principales ventajas y desventajas de cada una de las librerías se presentan a continuación.

Pandas

- **Ventajas**
  - Tiene estructuras de datos descriptivas, rápidas y compatibles.

- Soporta operaciones como agrupar, integrar, iterar, reindexar y representar datos.
- Es muy flexible para su uso en asociación con otras librerías de Python.
- Contiene funcionalidades inherentes a la manipulación de datos que pueden implementarse utilizando comandos mínimos.
- Se puede implementar en una gran variedad de áreas, especialmente relacionadas con los negocios y la educación, debido a su rendimiento optimizado.

- **Desventajas**
  - Es mucho menos adecuada para el modelado cuantitativo y matrices n-dimensionales. En estos escenarios, donde se necesita trabajar en modelado cuantitativo o estadístico, se utiliza NumPy.

Numpy

- **Ventajas**
  - Intuitivo e interactivo.
  - Ofrece transformadas de Fourier, capacidad de números aleatorios y otras herramientas para integrar lenguajes de programación como C, C++ y Fortran.
  - Puede tratar fácilmente con datos multidimensionales.
  - Ayuda en la manipulación matricial de datos y operaciones como la transposición, la matriz identidad y mucho más.
  - Permite mejorar el rendimiento y la gestión de la recolección de basura al proporcionar una estructura de datos dinámica.

- Permite mejorar el rendimiento de los modelos de “Machine Learning”.
- Otras librerías de “Machine Learning” como “Scikit Learn” y “TensorFlow” utilizan matrices NumPy como entrada.
- Tiene un gran apoyo y contribuciones de la comunidad de código abierto.
- Simplifica las implementaciones matemáticas complejas.

- **Desventajas**
  - Es altamente dependiente de entidades fuera de Python. Utiliza Cython y otras librerías que utilizar C o C++.
  - Sus tipos de datos son nativos de hardware y no nativos de Python, por lo que cuesta mucho cuando las entidades NumPy tienen que ser traducidas de nuevo a entidades equivalentes a Python y viceversa.

Matplotlib

- **Ventajas**
  - Soporta “shells” de Python y IPython, scripts de Python, Jupyter “Notebook”, servidores de aplicaciones web y muchos kits de herramientas de interfaz gráfica.
  - Opcionalmente, ofrece una interfaz parecida a la de MATLAB para realizar trazados sencillos.
  - La interfaz orientada a objetos ofrece un control completo de las propiedades de los ejes, las propiedades de las fuentes, los estilos de líneas y muchos más.

- Ayuda a producir gráficos que son configurables, potentes y precisos.
- Compatible con varios “backends” de gráficos y sistemas operativos.

- **Desventajas**
  - Curva de aprendizaje lenta.
  - No tiene identificadores protegido.

Scikit-learn

- **Ventajas**
  - Contiene un paquete completo de métodos que se pueden implementar para los algoritmos estándar de “Machine Learning”.
  - Su interfaz es sencilla y bien definida que permite realizar ajustes y trasformaciones a los modelos para cualquier colección de datos.
  - Es la mejor opción para la creación de pipelines, que permiten la construcción de un prototipo más rápido.
  - Es una de las mejores librerías para realizar un despliegue confiable de modelos de “Machine Learning”.
- **Desventajas**
  - Es muy útil para resolver problemas de aprendizaje supervisado, no se recomienda para problemáticas de tipo de aprendizajes no supervisado como lo es el “Deep Learning”.

2. Análisis exploratorio

Como se sabe, este es uno de los pasos más importantes en los proyectos de ciencia de datos, pues ayuda a entender los datos y a establecer si existe algún tipo de problema; desde allí, se determinan acciones para corregirlos o adecuarlos dependiendo de las necesidades identificadas.

Para entender mejor lo anteriormente dicho se desarrollará un ejemplo centrado en la creación de un modelo que permita la identificación de correos que pueden llegar a ser maliciosos.

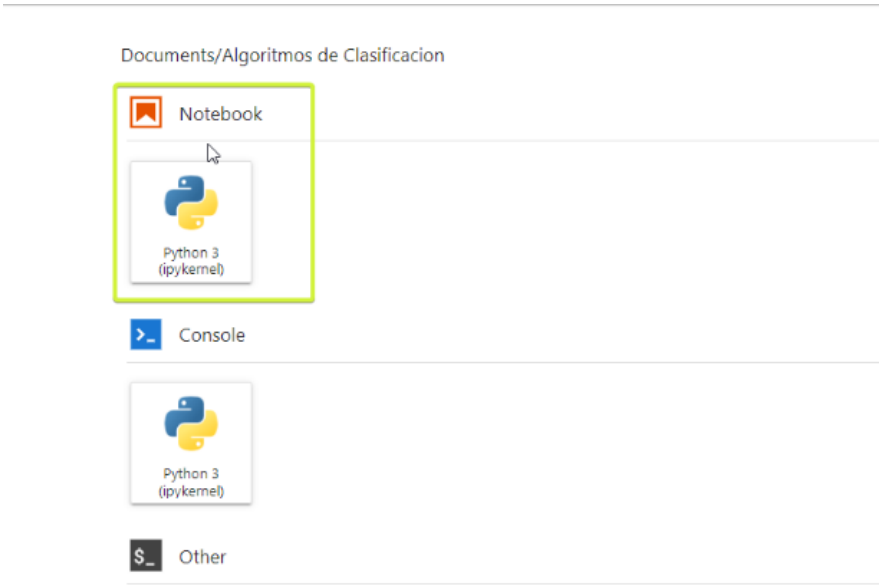
Se parte del hecho de que ya se ha realizado la búsqueda de la información y se cuenta con los insumos para empezar a trabajar.

En primer lugar, cabe suponer que se cuenta con una colección de datos correspondientes a mensajes de correos electrónicos que han sido catalogados como normales y maliciosos, de manera que se debe empezar con la exploración de los datos para ir entendiéndolos. Este proceso se realiza siguiendo unos pasos específicos, los cuales se presentan a continuación:

Proceso de exploración de datos

Paso 1. Entorno configurado: lo primero que se debe hacer es asegurarse de tener el entorno de trabajo configurado, en caso de no ser así, remítase a la primera parte de este componente y realice el proceso.

Paso 2. Abrir la aplicación JupiterLab de Anaconda y crear un nuevo Notebook. Observe la imagen.



Paso 3. Importar las librerías necesarias para la exploración de los datos, para nuestro ejemplo las librerías a utilizar en esta etapa son Pandas, Numpy, Matplolib. La importación se realiza utilizando los siguientes comandos

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

Paso 4. Se debe realizar la lectura de los archivos para el análisis: para el ejemplo trabajado el archivo en el cual están almacenados los datos es tipo CSV, y el comando necesario para realizar la lectura es la siguiente.

df=pd.read_csv('correosmaliciosos.csv')

pd: es el nombre de la librería que identifica a Pandas.

read_csv: función que permite leer datos en formato CSV.

correosmaliciosos.csv: archivo con los datos en formato CSV y ubicado en la misma carpeta del Notebook, en caso de estar ubicado en una ruta diferente deberá especificar la ruta completa, por ejemplo: 'C:/proyecto/correosmalicisos.csv', lo que indica que su ubicación esta en la carpeta proyecto de la unidad local C:

df= variable que almacena la información leída del archivo y que de denomina "dataframe", lo que significa que se tienen los datos en dos dimensiones determinados por filas y columnas, o instancias y atributos.

Paso 5. Una vez importados los datos, se procede con las primeras visualizaciones; para ello, podemos hacer usos de los siguientes comandos.

df

Solamente usando la variable o "dataframe" en el cual se guardaron los datos podemos visualizar los cinco primeros y cinco últimos registros de la colección de datos.

Utilice el comando **df.head()**, para visualizar los primeros 5 registros del conjunto de datos. En caso de requerir visualizar muchos más datos, coloque dentro del comando df.head(), el número de registros que quiera ver, por ejemplo, si necesita visualizar los 10 primeros use el comando de la siguiente manera **df.head(10)**

```
[4]: df
```

[4]:

	Unnamed: 0	fecha	salida	ciudad	mensaje
0	0	2022-05-30 15:48:58.221097	Normal	Pereira	Go until jurong point, crazy.. Available only ...
1	1	2022-05-30 15:48:58.221097	Normal	Bucaramanga	Ok lar... Joking wif u oni...
2	2	2022-05-30 15:48:58.221097	Malicioso	Medellin	Free entry in 2 a wkly comp to win FA Cup fina...
3	3	2022-05-30 15:48:58.221097	Normal	Medellin	U dun say so early hor... U c already then say...
4	4	2022-05-30 15:48:58.221097	Normal	Pereira	Nah I don't think he goes to usf, he lives aro...
...	...	...	...	...	...
5567	5567	2022-05-30 15:48:58.221097	Malicioso	Valledupar	This is the 2nd time we have tried 2 contact u...
5568	5568	2022-05-30 15:48:58.221097	Normal	Medellin	Will ü b going to esplanade fr home?
5569	5569	2022-05-30 15:48:58.221097	Normal	Cali	Pity, * was in mood for that. So...any other s...
5570	5570	2022-05-30 15:48:58.221097	Normal	Medellin	The guy did some bitching but I acted like i'd...
5571	5571	2022-05-30 15:48:58.221097	Normal	Medellin	Rofl. Its true to its name

5572 rows × 5 columns

Utilice el comando **df.head()**, para visualizar los primeros 5 registros del conjunto de datos.

```
[11]: df.shape

[11]: (5572, 5)
```

Paso 6. Entendiendo cada uno de los datos presentes en la colección de datos, para profundizar más sobre cada uno de los datos el comando que podemos utilizar es el siguiente, **df.info()**. Con este comando podemos determinar el tipo de dato, la totalidad de datos por cada atributo, de igual manera total genera de registros y columnas y totalizado por cada tipo de dato.

```
[13]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  5572 non-null   int64
 1   fecha       5572 non-null   object
 2   salida      5572 non-null   object
 3   ciudad      5572 non-null   object
 4   mensaje     5568 non-null   object
dtypes: int64(1), object(4)
memory usage: 217.8+ KB
```

Con el resultado se puede decir que tenemos una totalidad de 5572 registros y 5 columnas o atributos. Una columna de tipo numérica int64 denominada “Unnamed”: 0, 4 columnas de tipo “object”, denominadas fecha, salida, ciudad, y mensaje lo que quiere decir que son datos categóricos.

Paso 7. **Determinar datos nulos**: para realizar una revisión de todas las columnas determinando en cuales existe presencia de valores nulos o vacíos, utilice el comando, **df.isnull().sum()**, la función **isnull()**, ayuda a determinar cuántos valores son vacíos y con la función **sum()**, se realiza la suma de todos los valores nulos por cada columna, el resultado aplicado a nuestros datos es el siguiente.

```
[14]: df.isnull().sum()

[14]: Unnamed: 0    0
      fecha         0
      salida        0
      ciudad        0
      mensaje       4
      dtype: int64
```

El resultado nos muestra que en la columna mensaje hay una presencia de cuatro valores vacíos.

Paso 8. Análisis individual, veamos el contenido de las columnas que hacen parte de la colección de datos, por ejemplo, se realizan agrupaciones para determinar la participación porcentual y se analiza el contenido dentro de la columna salida. Con el siguiente comando se puede establecer los tipos de datos que están presentes en esa columna y cuánto es el total por cada característica.

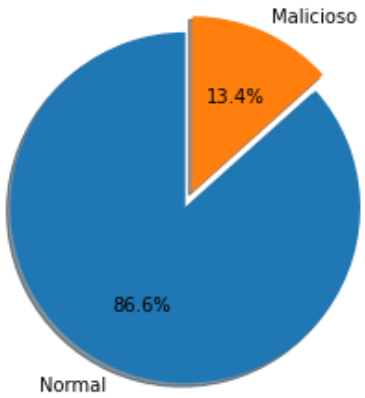
df['salida'.value_counts()]

```
[16]: df['salida'].value_counts()

[16]: Normal       4825
      Malicioso     747
      Name: salida, dtype: int64
```

Paso 9. Análisis gráfico, ahora veamos con el apoyo de los gráficos la participación porcentual de cada uno de valores, ejecutando el siguiente comando.

```
import matplotlib.pyplot as plt

etiquetas = 'Normal', 'Malicioso'

rebanada = (0, 0.1)

fig1, ax1 = plt.subplots()

ax1.pie(df['salida'].value_counts(), explode=rebanada, labels=etiquetas, autopct='%1.1f%%', shadow=True, startangle=90)

ax1.axis('equal')

plt.show()
```

El resultado que se debe obtener es el siguiente:



Paso 10. Análisis de resultados, se puede decir que con la exploración realizada se encontró la siguiente información:

Total de registros	5572
Columnas	5
Nombre de columnas	unanemed: 0 fecha salida ciudad mensaje
Valores nulos	Columna mensaje con 4 registros
Tipos de columna	1 numérica, 4 categóricas
Variable objetivo	Salida
Variables para el aprendizaje	fecha, ciudad, mensaje
Participación porcentual de la variable de salida	Normal (86.6%) Malicioso(13.4%)

3. Preprocesamiento de los datos con Python

Con el análisis exploratorio de los datos, se va haciendo una idea de cuáles serán las variables objetivas y de cuáles serán los atributos para el aprendizaje; de igual manera, se podrá determinar las correcciones necesarias que se deben realizar a los datos.

Se debe recordar que el objetivo es realizar un modelo que permita la detección de mensajes de correo electrónico y los clasifique como mensajes normales o maliciosos. Con esta premisa y el análisis que se ha realizado previamente se procede con los pasos para el preprocesamiento de los datos, tal y como se exponen a continuación.

Preprocesamiento de los datos

Paso 1. Definir los atributos que harán parte del análisis: para este caso, aunque se tienen cuatro columnas, el análisis se va a centrar en la columna mensaje como análisis para el entrenamiento, y como variable objetivo o resultante será la columna salida.

Lo que quiere decir que solamente se debe dejar en la colección de datos esas dos columnas, para realizar este proceso se debe ejecutar el siguiente comando.

```
df=df.drop(['Unnamed: 0', 'fecha', 'ciudad', axis=1)]
```

Este comando permite eliminar del "dataframe" las columnas que no se quieren analizar solamente dejando nuestra variable objetivo y las de análisis. Si se ejecuta el comando **df.head()**, se debe obtener el resultado de las primeras 5 filas, pero ya solamente de las dos columnas seleccionadas:

```
[27]: df=df.drop(['Unnamed: 0', 'fecha', 'ciudad'], axis=1)

[28]: df.head()
```

[28]:

	salida	mensaje
0	Normal	Go until jurong point, crazy.. Available only ...
1	Normal	Ok lar... Joking wif u oni...
2	Malicioso	Free entry in 2 a wkly comp to win FA Cup fina...
3	Normal	U dun say so early hor... U c already then say...
4	Normal	Nah I don't think he goes to usf, he lives aro...

Paso 2. Eliminar valores nulos: en la exploración de los datos se encontró que en la columna mensaje existen registros nulos, de manera que para este caso se procede con la eliminación de estos registros, aunque esta no es la única decisión.

En este caso son pocos registros y tomaremos la decisión de quitarlos de nuestra colección de datos.

Para proceder ejecute las siguientes líneas de comando.

df=df.dropna()

Con este comando se eliminarán todas las filas si en cualquiera de sus atributos o columnas encuentra un valor nulo. Recordemos que el total de registros era de 5572, una vez ejecutado el comando el resultado obtenido es el siguiente, con el comando "shape" rápidamente se puede ver el total de registros y el total de columnas.

```
[30]: df=df.dropna()

[32]: df.shape

[32]: (5568, 2)
```

Paso 3. Para el ejemplo la variable objetivo es un texto de tipo categórico y es necesario que se realice su transformación para que puedan ser procesados por los algoritmos de clasificación que implementaremos.

La variable que se utilizará para realizar el aprendizaje corresponde a los mensajes y para realizar la adecuación de estos es la librería nltk.

import nltk

nltk.download("punkt")

import warnings

warnings.filterwarnings('ignore')

from wordcloud import WordCloud

Como primer análisis de los mensajes se creará una nube de palabras que permitan identificar cuáles de ellas hacen parte de los mensajes normales y cuáles de los mensajes maliciosos.

Primero se debe dos variables vacías de la siguiente manera.

palabrasNormales = ''

palabraMaliciosos = ''

Paso 4. Posteriormente se debe recorrer el "dataframe" en busca de los mensajes que coincidan con las etiquetas Malicioso, de la siguiente manera.

```
for mensaje in df[df['salida'] == 'Malicioso'].mensaje:

texto = mensaje.lower()

tokens = nltk.word_tokenize(texto)

for palabras in tokens:

palabrasMalicioso = palabrasMalicioso + palabras + ' '
```

for mensaje in df[df['salida'] == 'Malicioso'].mensaje: recorre cada elemento en los cuales la etiqueta de la columna salida coincide con Malicioso.

texto = mensaje.lower(): convierte cada una de las coincidencias en minúscula.

tokens = nltk.word_tokenize(texto): separa el mensaje en una lista de todas las palabras que la conforman.

for palabras in tokens: recorre cada una de las palabras para ser almacenadas en una sola cadena de texto.

Paso 5. El objetivo de implementar dicho código es tomar todos los mensajes que coincidan con la etiqueta de salida malicioso, convertirlas a minúsculas para estandarizar todos los textos, luego separar cada palabra en una sola cadena de texto, que servirá para analizar cuál de las palabras tienen mayor representación.

Una vez con todas las palabras en una sola cadena de texto se procede a generar la nube de palabras.

cloudMalicioso = WordCloud(width=500, height=300, background_color='white').generate(palabrasMalicioso)

La variable la denominamos cloudMalicioso y con la librería WordCloud, generarmos la nube de palabras con la variable que previamente ya hemos usado para tal fin denominada palabrasMalicioso.

Se procede con la grafica implementando los siguientes códigos.

plt.figure(figsize=(10,8))

plt.imshow(cloudMalicioso)

plt.axis("off")

plt.tight_layout(pad=0)

plt.show()

El resultado de ejecutar las siguientes líneas de código será la representación de nube de palabras como se aprecia en la imagen.

Paso 6. Se realiza el mismo procedimiento, pero esta vez detectando los mensajes de los correos que están marcados como Normal, el código quedaría de la siguiente manera.

```
for mensaje in df[df['salida'] == 'Normal'].mensaje:

texto = mensaje.lower()

tokens = nltk.word_tokenize(texto)

for palabras in tokens:

palabrasNormal = palabrasNormal + palabras + ' '

cloudNormal = WordCloud(width=500, height=300,

background_color='white').generate(palabrasNormal)

plt.figure( figsize=(10,8))

plt.imshow(cloudNormal)

plt.axis("off")

plt.tight_layout(pad=0)

plt.show()
```

Y la nube de palabras que se obtiene para este caso es la siguiente:

Paso 7. Ahora se deben convertir las etiquetas de salida: para que la máquina entienda los datos se procede con las transformaciones de las etiquetas normal y

malicioso, la estrategia sería asignarle un número a cada una de estas de la siguiente manera:

Etiqueta	Valor
Normal	0
Malicioso	1

Para realizar la transformación de estos datos ejecute la siguiente línea de comandos.

df = df.replace(['Normal','Malicioso'],[0, 1])

df.head(10)

El resultado se puede observar en la imagen

```
[30]: df = df.replace(['Normal','Malicioso'],[0, 1])
      df
```

[30]:

	salida	mensaje
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...
5568	0	Will ü b going to esplanade fr home?
5569	0	Pity, * was in mood for that. So...any other s...
5570	0	The guy did some bitching but I acted like i'd...
5571	0	Rofl. Its true to its name

Como se aprecia en la columna salida ya se visualiza la información correspondiente a 0 para la etiqueta Normal y 1 para la etiqueta Malicioso.

Paso 8. Ahora se debe eliminar la información que no aporta nada para el modelo, de tal manera que se procede con la eliminación de los signos de puntuación, y de las palabras vacías, para ello se hace uso de la librería NTKL.

```
import nltk

from nltk.corpus import stopwords

import string

def eliminarPuntuacion(texto):

texto = texto.translate(str.maketrans('', '', string.punctuation))

texto = [word for word in texto.split() if word.lower() not in

stopwords.words('english')]

return " ".join(texto)

df['mensaje'] = df['mensaje'].apply(eliminarPuntuacion)

df
```

El resultado se puede ver comparando los resultados antes y después de la depuración:

[40]:

	salida	mensaje
0	0	Go jurong point crazy Available bugis n great ...
1	0	Ok lar Joking wif u oni
2	1	Free entry 2 wkly comp win FA Cup final tkts 2...
3	0	U dun say early hor U c already say
4	0	Nah dont think goes usf lives around though
...	...	...
5567	1	2nd time tried 2 contact u U £750 Pound prize ...
5568	0	ü b going esplanade fr home
5569	0	Pity mood Soany suggestions
5570	0	guy bitching acted like id interested buying s...
5571	0	Rofl true name

5568 rows × 2 columns

4. Crear el modelo

Después de la exploración y el preprocesamiento, se continúa con la construcción del modelo, como se puede ver, la variable con la que aprenderá el modelo se basa en texto que ya se ha dividido en palabras y que se debe empezar a cuantificar para poder pasar la información para que el modelo lo pueda entender.

Como se han venido desarrollando todas las unidades se realizará una serie de pasos que permitan en este caso la construcción de un modelo:

Paso 1. Separación de datos. Se deben separar los datos en los datos entrenamiento y las clases de la siguiente manera, el "dataframe" mensaje corresponde a las instancias y atributos, mientras que las etiquetas a las clases.

```
mensaje = pd.DataFrame(df['mensaje'])

etiquetas = pd.DataFrame(df['salida'])

print(mensaje.head())

print(etiquetas.head())
```

El resultado es un "dataframe" independiente en el que se dividen los datos de aprendizaje de las clases, como se observa en la imagen.

```
[43]: mensaje = pd.DataFrame(df['mensaje'])
      etiquetas = pd.DataFrame(df['salida'])

[45]: print(mensaje.head())
      print(etiquetas.head())
```

```
                                             mensaje
0  Go jurong point crazy Available bugis n great ...
1                            Ok lar Joking wif u oni
2  Free entry 2 wkly comp win FA Cup final tkts 2...
3                U dun say early hor U c already say
4       Nah dont think goes usf lives around though
   salida
0       0
1       0
2       1
3       0
4       0
```

Paso 2. Conversión de palabras a vectores. Lo que se quiere con este procedimiento es realizar el recuento de los "tokens" o palabras que se obtienen de los mensajes de correo electrónico, el objetivo es crear una matriz que realice un recuento por cada una de las palabras y las construya como un atributo. Para obtener este resultado debe ejecutar los siguientes comandos.

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer() vector = vectorizer.fit_transform(df['mensaje'])

vector.shape

El resultado que se obtiene lo puede observa en la imagen.

```
[70]: from sklearn.feature_extraction.text import TfidfVectorizer

      vectorizer = TfidfVectorizer()
      vector = vectorizer.fit_transform(df['mensaje'])
      vector.shape

[70]: (5568, 9426)
```

Paso 3. Objetivo encontrado. Ya se ha logrado cuantificar cada una de las palabras del texto de tal manera que estos serán los datos con los que entrenaremos el modelo. Lo que resta es almacenar esta variable a una que servirá para el siguiente paso del entrenamiento, para el ejemplo la denominaremos instancias.

4.1. Entrenamiento

Se puede decir que se tiene la base para empezar a realizar el entrenamiento de los datos, toda vez que ya se sabe cuáles son los datos de entrada y las clases, y ambas se encuentran en valores numéricos tal como se requiere. En el siguiente recurso se dan a conocer algunas recomendaciones importantes para realizar el entrenamiento:

- **Selección de datos**. Seleccionar los datos de manera aleatoria. No se deben escoger datos que sean continuos o de manera organizada.
- **Separar los datos**. Reserve una muestra que por lo general equivale al veinte por ciento de los datos, los cuales debe asegurarse que no se utilizaron en el aprendizaje.
- **Entrenar los datos**. Seleccione el ochenta por ciento de los datos para realizar el entrenamiento de los datos.

- **Selección de algoritmo**. De acuerdo con el requerimiento, seleccione el algoritmo o algoritmos de clasificaciones que mejor le vayan dependiendo del tipo de problema a resolver.

Para realizar el entrenamiento seguir cada uno de los siguientes pasos:

Paso 1. Como primer paso se procede a seleccionar los datos de entrenamiento y de prueba, para ello haga usos de los siguientes comandos.

```
from sklearn.model_selection import
GridSearchCV,train_test_split,StratifiedKFold,cross_val_score,learning_curve

X_train, X_test, y_train, y_test = train_test_split(instancias, df['salida'],
test_size=0.2, random_state=131)
```

- **X_train:** corresponde a los datos de entrenamiento.
- **X_test:** Aquí se guardan los datos que son separados para realizar las pruebas.
- **y_train:** etiquetas de salida correspondientes a las instancias de entrenamiento.
- **y_test:** etiquetas de salida que corresponde a las instancias seleccionadas de entrenamiento.
- **instancias:** datos preparados para el entrenamiento.
- **df['salida':]** etiquetas o clases de salida de la clasificación de las instancias.
- **test_size:** valor entre 0 y 1 que indica cuál es el porcentaje de datos de prueba que serán separados, para nuestro ejemplo será del 20 por ciento, es decir 0.2.

- **random_state:** este parámetro se utiliza para realizar la mezcla de los datos aleatoriamente tal como se recomienda anteriormente, el número corresponde al número de llamados de la función y la mezcla se realiza antes de la división de los datos.

Paso 2. Lo que sigue es la selección del algoritmo de clasificación, en este caso se utilizará el de "Naive Bayes", y Regresión Logística.

Se importan las librerías necesarias para poder implementarlas.

from sklearn.naive_bayes import MultinomialNB

from sklearn.linear_model import LogisticRegression

Para utilizar el algoritmo de "Naive Bayes", se le asigna la configuración a la variable NB,

NB = MultinomialNB(alpha=0.5)

Tener en cuenta el siguiente parámetro.

alpha= parámetro de suavizado de Laplace si se omite por defecto asignaría 1, para nuestro caso se establecerá en 0.5

Para el algoritmo de regresión logística se utilizará esta configuración, que se almacenará en la variable RL.

RL = LogisticRegression(solver='liblinear', penalty='l1')

Solver: algoritmo para utilizar en el problema de optimización. El valor predeterminado es 'lbfgs'. Para elegir un solucionador, es posible que se desee considerar los siguientes aspectos:

- Para conjuntos de datos pequeños, 'liblinear' es una buena opción, mientras que 'sag' y 'saga' son más rápidos para conjuntos grandes;
- Para problemas multiclase, solo 'newton-cg', 'sag', 'saga' y 'lbfgs' manejan la pérdida multinomial;
- 'liblinear' se limita a esquemas de uno contra el resto.

Paso 3. Se utilizan los siguientes comandos para realizar el entrenamiento con el algoritmo de "Naive Bayes" usar los siguientes comandos.

NB.fit(X_train,y_train)

Para realizar el entrenamiento con el algoritmo de Regresión Logística, de igual manera use el mismo comando, pero con su respectiva configuración almacenada en la variable RL

RL.fit(X_train,y_train)

Todos estos pasos se ven unificados de la siguiente manera:

- Importación de las librerías:

  from sklearn.naive_bayes import MultinomialNB

  from sklearn.linear_model import LogisticRegression
- Configuración de los parámetros de los algoritmos de clasificación:

  NB = MultinomialNB(alpha=0.2)

  RL = LogisticRegression(solver='liblinear', penalty='l1')
- Entrenamiento:

  NB.fit(X_train,y_train)

  RL.fit(X_train,y_train)

4.2. Evaluación

En esta etapa del proceso se debe realizar la evaluación de los modelos. Una vez entrenados se debe probar qué tan eficientes han sido dichos modelos, es importante recordar que en principio se separó un conjunto de datos correspondientes al veinte por ciento, y son estos los que ayudarán a probar el modelo y ver qué tan eficiente es.

Para el proceso de evaluación se deben seguir estos pasos:

Paso 1. Se deben importar las librerías necesarias para realizar las métricas.

```
from sklearn import metrics
```

Paso 2. Realizar la predicción de los datos con el modelo entrenado, primero con "Naive Bayes" y posteriormente con regresión logística, recordemos que los modelos son NB y RL respectivamente; para realizar la predicción, utilice los datos de prueba y ejecute el siguiente comando.

```
y_pred_NB = NB.predict(X_test)
```

Los resultados se almacenan en y_pred_NB, y para el modelo de regresión logística realice el mismo procedimiento, pero almacenando en una variable diferente.

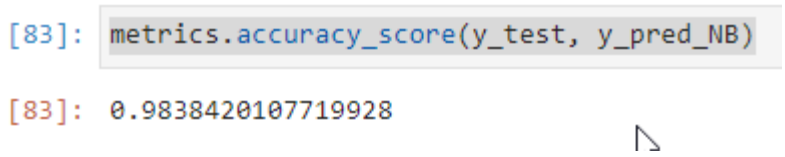
```
y_pred_RL = RL.predict(X_test)
```

Paso 3. Las salidas de la predicción del modelo entrenado se deben comparar con las salidas correctas para ese conjunto de datos, y de esta manera saber el porcentaje de aciertos para cada uno de los modelos. La comparativa se realiza implementando las siguientes líneas de código.

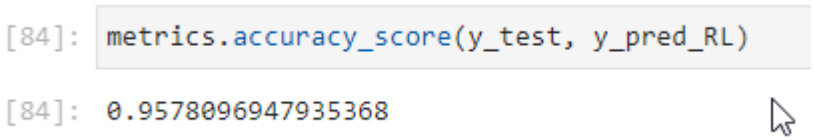
Para el modelo de "Naive Bayes", utilice el siguiente comando:

metrics.accuracy_score(y_test, y_pred_NB)

Y el resultado que se obtiene se aprecia en la siguiente imagen.



Paso 4. De igual manera se debe comparar en el modelo de regresión logística, cuál es resultado obtenido, ejecutando el mismo comando así:



Como se indicó en las recomendaciones, se dejaron los datos para las pruebas y efectivamente sirvieron para validar la efectividad del modelo, el resultado arrojado es del 98 % donde se utilizó el algoritmo de clasificación “Naive Bayes”, mientras que con el modelo donde se implementó regresión logística se obtuvo un porcentaje de acierto del 95 %.

4.3. Predicciones

Con los modelos evaluados se procede a realizar las predicciones, esta vez con entradas completamente nuevas. Con el aprendizaje realizado el modelo debería estar en la capacidad de clasificar los datos en correos electrónicos normales o maliciosos.

Es importante aclarar que los nuevos datos que le pasen al modelo deben estar relacionados con la temática bajo la cual este modelo fue entrenado, o sea con mensajes de correos electrónicos.

Para realizar las predicciones se tomará el modelo con mayor efectividad que arrojó la evaluación, que sería el que se entrenó con el algoritmo de clasificación "Naive Bayes" con un porcentaje del 98 por ciento.

Se establecerán múltiples entradas y para este fin se deben seguir los pasos expuestos:

Paso 1. Definir las entradas.

```
Entrada1=['free mobile now']

Entrada2=['Sorry Mr James']

Entrada3=['I need your help']

Entrada4=['call me text']
```

Paso 2. Crear una función para realizar las predicciones, de la siguiente manera

```
def clasificar(resultado):

if resultado == 1:

print ("Este es un Mensaje Malicioso")

else: print ("Es un Mensaje Normal")
```

La función espera por un resultado en caso de ser 1 lo clasifica como mensaje malicioso de lo contrario como mensaje normal.

Paso 3. Vectorizar las entradas nuevas, así:

salida=vectorizer.transform(Entrada1)

Se deben transformar los datos y dejarlos de igual manera de como se entrenaron.

Paso 4. Predecir y mostrar el resultado. Realizar la predicción con la salida de los datos vectorizados y evaluar el resultado con la función de clasificación. resultado = NB.predict(salida) print('Mensaje:',Entrada1) clasificar(resultado) El resultado para la primera entrada se aprecia en la imagen.

```
[99]: Entrada1=['free mobile now']
      Entrada2=['Sorry Mr James']
      Entrada3=['I need your help']
      Entrada4=['call me text']

[96]: def clasificar(resultado):
          if resultado == 1:
              print ("Este es un Mensaje Malicioso")
          else:
              print ("Es un Mensaje Normal")

[100]: salida=vectorizer.transform(Entrada1)
       resultado = NB.predict(salida)
       print('Mensaje: ',Entrada1)
       clasificar(resultado)

       Mensaje:  ['free mobile now']
       Este es un Mensaje Malicioso

[ ]:
```

Paso 5. Predecir más entradas. Realizar más pruebas para verificar la efectividad del modelo, las demás entradas de nuestro ejemplo dan como resultado la información de la imagen.

[102]:
```
salida=vectorizer.transform(Entrada1)
resultado = NB.predict(salida)
print('Mensaje 1: ',Entrada1)
clasificar(resultado)

salida=vectorizer.transform(Entrada2)
resultado = NB.predict(salida)
print('Mensaje 2: ',Entrada2)
clasificar(resultado)

salida=vectorizer.transform(Entrada3)
resultado = NB.predict(salida)
print('Mensaje 3: ',Entrada3)
clasificar(resultado)

salida=vectorizer.transform(Entrada4)
resultado = NB.predict(salida)
print('Mensaje 4: ',Entrada4)
clasificar(resultado)
```

```
Mensaje 1:  ['free mobile now']
Este es un Mensaje Malicioso
Mensaje 2:  ['Sorry Mr James']
Es un Mensaje Normal
Mensaje 3:  ['I need your help']
Es un Mensaje Normal
Mensaje 4:  ['call me text']
Es un Mensaje Normal
```

Paso 6. Finalmente, el código completo del proceso de predicción es el siguiente.

Entrada1=['free mobile now']

Entrada2=['Sorry Mr James']

Entrada3=['I need your help']

Entrada4=['call me text']

def clasificar(resultado):

if resultado == 1:

```
print ("Este es un Mensaje Malicioso")

else:

print ("Es un Mensaje Normal")

salida=vectorizer.transform(Entrada1)

resultado = NB.predict(salida)

print('Mensaje 1: ',Entrada1)

clasificar(resultado)

salida=vectorizer.transform(Entrada2)

resultado = NB.predict(salida)

print('Mensaje 2: ',Entrada2)

clasificar(resultado)

salida=vectorizer.transform(Entrada3)

resultado = NB.predict(salida)

print('Mensaje 3: ',Entrada3)

clasificar(resultado)

salida=vectorizer.transform(Entrada4)

resultado = NB.predict(salida)

print('Mensaje 4: ',Entrada4)
```

4.4. Evaluar el modelo y análisis de errores

Finalmente, para analizar el modelo en cuanto al número de errores que se pueden presentar, ya sea porque se realiza la clasificación de correos electrónicos maliciosos como normales, o viceversa, se utilizará la técnica denominada matriz de confusión.

Para implementar dicha técnica se deben seguir los pasos expuestos a continuación:

- **Paso 1. Importar las librerías**

Para realizar la implementación usaremos la librería que nos ayuda con el proceso, que es la primera, y la segunda librería que nos permitirá las visualizaciones gráficas.

```
from sklearn.metrics import confusion_matrix

import seaborn as sns
```

- **Paso 2. Realizar la predicción con las pruebas**

Procedemos nuevamente a realizar las predicciones con nuestro conjunto de datos de prueba, ya que son los indicados para generar un análisis que permita determinar qué tantos errores se presentan en nuestro modelo.

Para realizar este procedimiento utilice las siguientes líneas de código que le permitirán calcular las predicciones con el modelo y también asignar las predicciones correctas.

y_pred_NB = NB.predict(X_test)

y_correctas_NB = y_test

Como se aprecia, la primera línea corresponde a las predicciones con el modelo entrenado usando el algoritmo "Naive Bayes", pasándole como entradas los datos de prueba que fueron separados, y la segunda línea simplemente le reasignamos a una nueva variable los valores de salida correcto de los datos de prueba.

- **Paso 3. Analizar los errores**

Con los datos de las salidas de las predicciones y las salidas correctas se aplica el método de matriz de confusión, de la siguiente manera.

MC = confusion_matrix(y_correctas_NB, y_pred_NB)

- **Paso 4. Graficar resultados**

Implemente las salidas de manera gráfica para entender mucho mejor los datos resultantes; para ello, utilice los siguientes comandos.

f, ax = plt.subplots(figsize =(5,5))

sns.heatmap(MC,annot = True,linewidths=0.5,linecolor="blue",fmt = ".0f",ax=ax)

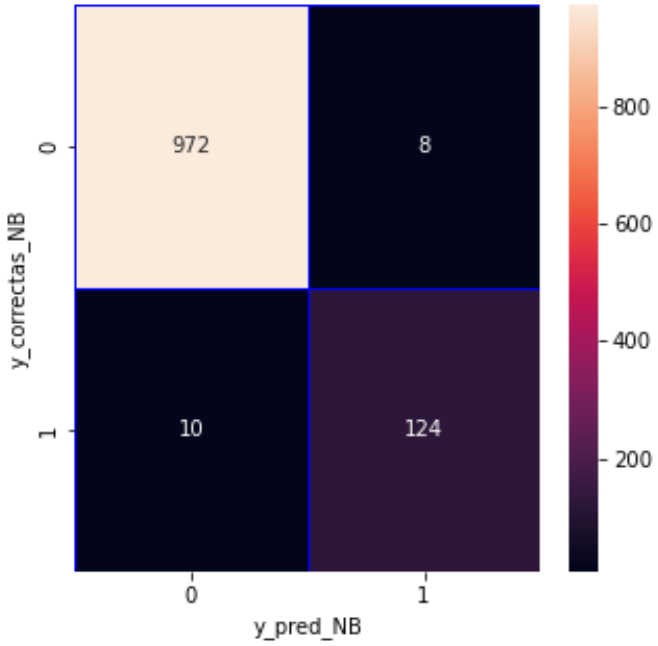
plt.xlabel("y_pred_NB")

plt.ylabel("y_correctas_NB")

plt.show()

- **Paso 5. Analizar gráfico**

El resultado que arroja la técnica de matriz de confusión se puede observar en la siguiente gráfica, en la cual existe un equilibrio, ya que las veces en las que predijo correos normales como maliciosos es de 10, y en las que predijo correos maliciosos como normales de 8.



- **Paso 6. Otros cálculos**

Para realizar los demás cálculos utilice los siguientes comandos.

- Para calcular la exhaustividad utilice:

from sklearn.metrics import recall_score

recall_score(y_correctas_NB, y_pred_NB, average=None)

Resultado: array([0.99183673, 0.92537313])

- Para calcular la precisión utilice:

  from sklearn.metrics import

  precision_score

  precision_score(y_correctas_NB, y_pred_NB, average=None)

  Resultado: array([0.9898167 , 0.93939394])

- Para el cálculo de la métrica F1

  from sklearn.metrics import f1_score

  f1_score(y_correctas_NB, y_pred_NB, average=None)

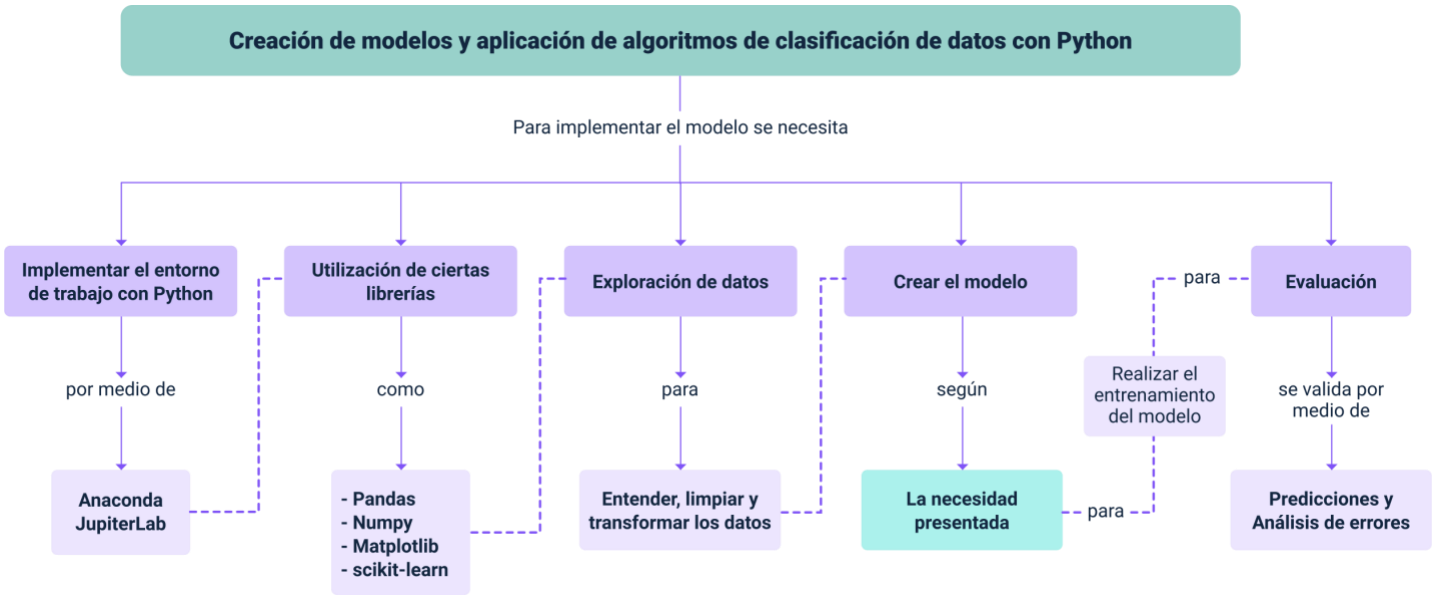
  Resultado: array([0.99082569, 0.93233083])

Se ha llegado al final de este espacio de formación en el que se ha diseñado un modelo de aprendizaje supervisado utilizando dos algoritmos de clasificación, "Naive Bayes" y el de regresión logística, aunque los dos dieron muy buenos resultados, el que más precisión obtuvo fue el modelo en el cual se utilizó "Naive Bayes", con un 98 por ciento de aciertos, lo que lo convierte en un algoritmo ideal para este tipo de problemas donde las variables de salida son de tipo binario.

Síntesis

Es innegable que las empresas deben buscar estrategias para organizar cada uno de sus procesos, y esto implica el implementar modelos que les permitan organizar, clasificar y hacer usos de los datos para tomar decisiones adecuadas que lleven a cumplir los objetivos establecidos por la organización. Este espacio de formación está destinado a conocer, mediante ejemplos específicos, los pasos para realizar el análisis de información utilizando Anaconda, sus diferentes librerías y el modelo "Machine Learning" con Python.

Una breve revisión de los temas vistos, se encuentran en el siguiente esquema:



Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
4. Crear el modelo	SDC LEARNING. (2022). Webinar Gratuito Mi Primer Modelo de Machine Learning en Python. [Video]. YouTube.	Video	https://www.youtube.com/watch?v=9HKfqinJJAo
4. Crear el modelo	Parra, F. (2022). 6 Métodos de clasificación | Estadística y Machine Learning con R. Algoritmo de Clasificación.	Página web	https://bookdown.org/content/2274/metodos-de-clasificacion.html
4.4. Evaluar el modelo y análisis de errores	González, L. (2020). Evaluando el error en los modelos de clasificación. Aprende IA.	Página web	https://aprendeia.com/evaluando-el-error-en-los-modelos-de-clasificacion-machine-learning/

Glosario

Algoritmos de clasificación: hacen parte del aprendizaje supervisado y se caracterizan porque los modelos se entrenan con las instancias y sus respectivas clasificaciones.

Entrenamiento: proceso que se realiza para que los modelos aprendan de los datos.

Evaluación: análisis de eficiencia con el que el modelo predice los datos, generalmente se contrasta con una colección de pruebas separada previamente.

Matriz de confusión: es una métrica para establecer el nivel de error, precisión y otras medidas en los modelos de "Machine Learning".

"Naive Bayes": algoritmo de clasificación utilizado en problemas donde las salidas son binarias.

Predicciones: capacidad del modelo para clasificar entradas nuevas, de acuerdo con un entrenamiento previo.

Preprocesamiento: manipulación que se realiza a los datos con el objetivo de entregarlos al modelo como este lo requiera.

Scikit-learn: librería especializada para proyectos de "Machine Learning" y ampliamente utilizada por todo el mundo.

Referencias bibliográficas

González, L. (2020). Regresión Logística- Teoría. Aprende IA. https://aprendeia.com/algoritmo-regresion-logistica-machine-learning-teoria/#:~:text=La%20Regresi%C3%B3n%20Lineal%20proporciona%20una,el%20precio%20de%20una%20acci%C3%B3n

Miller, V. (2020). Explorando Algoritmos de Aprendizaje Automático Supervisado. Toptal Engineering Blog. https://www.toptal.com/machine-learning/explorando-algoritmos-de-aprendizaje-automatico-supervisado

Román, V. (2019). Machine Learning: Cómo Desarrollar un Modelo desde Cero. Medium https://medium.com/datos-y-ciencia/machine-learning-c%C3%B3mo-desarrollar-un-modelo-desde-cero-cc17654f0d48

Román, V. (2019). Aprendizaje Supervisado: Introducción a la Clasificación y Principales Algoritmos. Medium. https://medium.com/datos-y-ciencia/aprendizaje-supervisado-introducci%C3%B3n-a-la-clasificaci%C3%B3n-y-principales-algoritmos-dadee99c9407

SMS Spam Collection Dataset. (s.f.). Kaggle https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

Sotaquirá, M. (2021). ¿Se requiere SQL para trabajar en Machine Learning? Codificando Bits https://www.codificandobits.com/blog/sql-machine-learning/

Créditos

Nombre	Cargo	Regional y Centro de Formación
Claudia Patricia Aristizábal	Responsable del Ecosistema	Dirección General
Rafael Neftalí Lizcano Reyes	Responsable de Línea de Producción	Regional Santander - Centro Industrial del Diseño y la Manufactura
Ronald Alexander Vacca Ascanio	Experto Temático	Regional Santander - Centro Industrial del Diseño y la Manufactura
Jeimy Lorena Romero Perilla	Diseñador Instruccional	Regional Norte de Santander - Centro de la industria, la empresa y los servicios
Carolina Coca Salazar	Asesora Metodológica	Regional Distrito Capital - Centro de Diseño y Metrología
Jhon Jairo Rodríguez Pérez	Corrector de Estilo	Regional Distrito Capital - Centro de Diseño y Metrología
Miroslava González Hernández	Diseñadora Instruccional	Regional Santander - Centro Industrial del Diseño y la Manufactura
Juan Daniel Polanco	Diseñador de Contenidos Digitales	Regional Santander - Centro Industrial del Diseño y la Manufactura
Camilo Andrés Bolaño Rey	Desarrollador FullStack	Regional Santander - Centro Industrial del Diseño y la Manufactura
Camilo Andrés Bolaño Rey	Locución	Regional Santander - Centro Industrial del Diseño y la Manufactura
Daniela Muñoz Bedoya	Animador y Producción Audiovisual	Regional Santander - Centro Industrial del Diseño y la Manufactura
Emilsen Bautista	Actividad Didáctica	Regional Santander - Centro Industrial del Diseño y la Manufactura

Nombre	Cargo	Regional y Centro de Formación
Zuleidy María Ruiz Torres	Validador de Recursos Educativos Digitales	Regional Santander - Centro Industrial del Diseño y la Manufactura
Luis Gabriel Urueta Álvarez	Validación de Recursos Educativos Digitales	Regional Santander - Centro Industrial del Diseño y la Manufactura
Daniel Ricardo Mutis Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Regional Santander - Centro Industrial del Diseño y la Manufactura