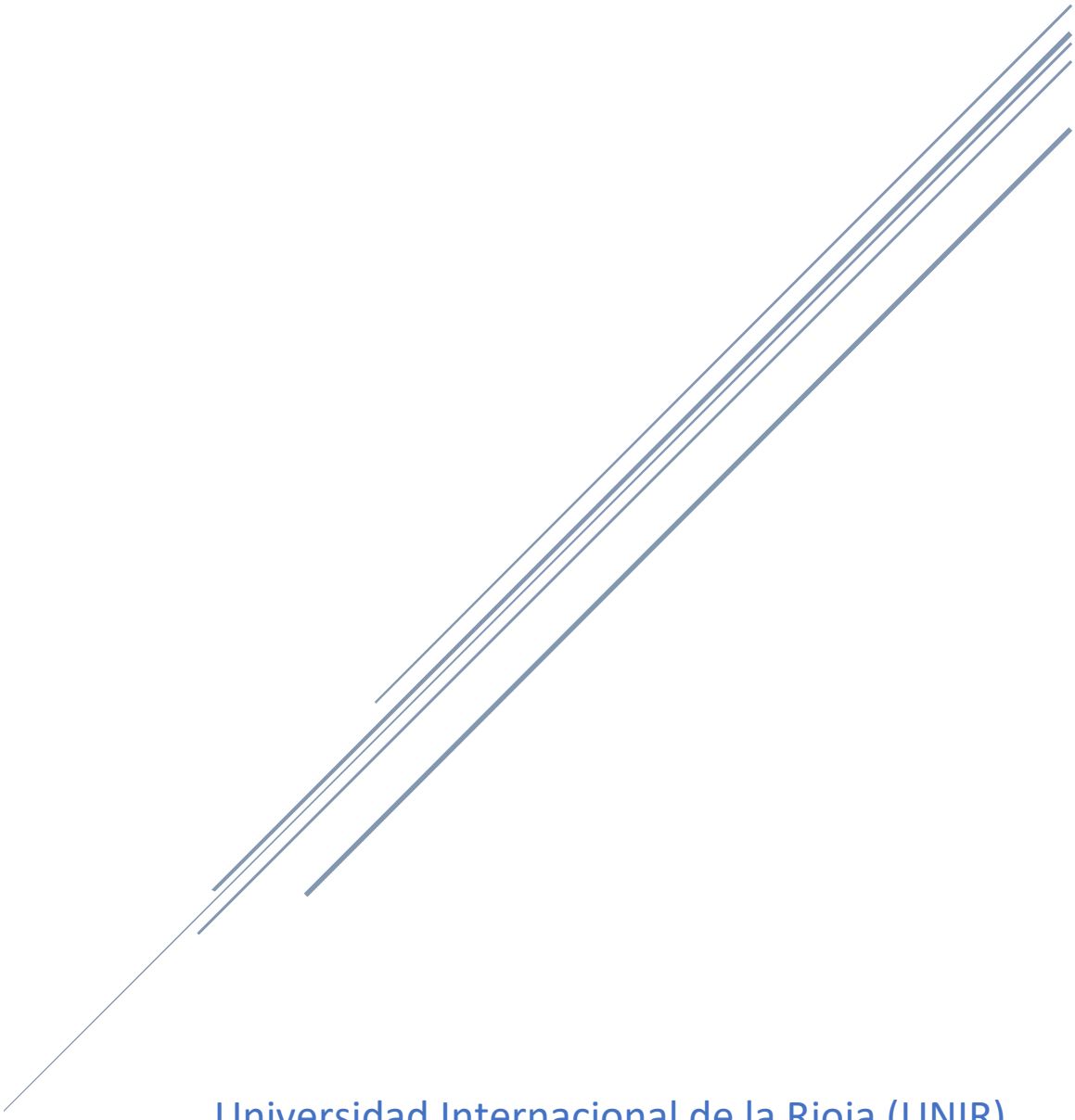
ACTIVIDAD 1

Limpieza de Dataset



Universidad Internacional de la Rioja (UNIR)
Métodos de captura y almacenamiento de datos

Asignatura	Datos del alumno	Fecha
Métodos de Captura y Almacenamiento de los Datos	Apellidos: Siguantay González	06-06-2021
	Nombre: Wilder Emmanuel	

Contents

Asignatura	Datos del alumno	Fecha
Métodos de Captura y Almacenamiento de los Datos	Apellidos: Siguantay González	06-06-2021
	Nombre: Wilder Emmanuel	

Descripción del problema.

El problema que pude encontrar en este Dataset son dos, específicamente con el formato del archivo CSV, el cual está separado por punto y coma y no por coma como normalmente se trabaja, cuando se lee este archivo con la librería de Pandas en Python, lo lee todo como una sola columna, una vez solucionado este inconveniente, el otro problema son registros duplicados, estos registros también los encontré con una función de Pandas, puntualmente son 6 registros duplicados (se verificó por CrimeId) de los cuales dejo la siguiente lista:
[160913455, 160913455, 160913455, 160950496, 160950496, 160950496]

Campos Nulos

También cabe resaltar, que se comprobó el número de valores nulos de cada columna.

```
df.isnull().sum() #Verificar nulos y contarlos.
```

```
CrimeId                     0
OriginalCrimeTypeName       0
OffenseDate                 0
CallTime                    0
CallDateTime                0
Disposition                 0
Address                     0
City                      321
State                       3
AgencyId                    0
Range                   10051
AddressType                 0
dtype: int64
```

Campos redundantes

Hay campos en el Dataset que son redundantes, por ejemplo, los datos de OffenseDate y el CallTime también lo tenemos en CallDateTime

OffenseDate	CallTime	CallDateTime

30/03/2016 00:00	06:42:00 p. m.	30/03/2016 18:42
31/03/2016 00:00	03:31:00 p. m.	31/03/2016 15:31
31/03/2016 00:00	04:49:00 p. m.	31/03/2016 16:49

El registro de Range, en realidad no representa nada, ya que todos los campos están vacíos

Range

Ejemplo del archivo CSV original:

CrimeId;OriginalCrimeTypeName;OffenseDate;CallTime;CallDateTime;Disposition;Address;City;State;AgencyId;Range;AddressType

160903280;Assault / Battery;2016-03-30T00:00:00;18:42;2016-03-30T18:42:00;REP;100 Block Of Chilton Av;San Francisco;CA;1;;Premise Address

Ejemplo de registros duplicados extraídos del CSV

CrimeId	OriginalCrimeTypeName	OffenseDate	CallTime	CallDateTime	Disposition	Address
160913455	Vandalism	31/03/2016 00:00	08:53:00 p. m.	31/03/2016 20:53	ND	1600 Block Of Sunnydale Av
160913455	Susp	01/04/2016 00:00	06:29:00 p. m.	01/04/2016 18:29	GOA	Geary St/larkin St
160913455	Passing Call	02/04/2016 00:00	05:11:00 p. m.	02/04/2016 17:11	Not recorded	900 Block Of Market St
160950496	Passing Call	04/04/2016 00:00	06:51:00 a. m.	04/04/2016 06:51	HAN	University St/felton St
160950496	Suspicious Vehicle	04/04/2016 00:00	06:51:00 a. m.	04/04/2016 06:51	ND	1400 Block Of Cabrillo St
160950496	Trespasser	04/04/2016 00:00	06:51:00 a. m.	04/04/2016 06:51	CAN	Block Of Hampshire St

Asignatura	Datos del alumno	Fecha
Métodos de Captura y Almacenamiento de los Datos	Apellidos: Siguantay González	06-06-2021
	Nombre: Wilder Emmanuel	

Solución

Solución al archivo CSV.

La solución más sencilla en este caso, sería cambiar el delimitador que tiene originalmente (“;”) por el que se usa por defecto para este tipo de archivos (“,”), también se puede colocar que tipo de delimitador tendrá nuestro archivo a la hora de leerlo, pero, la mayoría de programas y lenguajes de programación (Excel, Java, Python, etc.) utilizan la coma por defecto, por lo que cada vez que se lea se tendría que estar indicando el delimitador que se utiliza, en mi opinión, es mejor dejarle el delimitador por defecto para hacer una lectura más rápida sin indicar su delimitador cada vez que se quiera leer.

Solución a registros duplicados

Como podemos ver, los registros “duplicados” que se encontraron anteriormente, en realidad no están duplicados, sino lo único duplicado que tienen es en número de **CrimeId**, por lo que podemos concluir que no son duplicados en su totalidad. Al verificar que la información de esas filas no es exactamente igual, conviene mantener los registros, lo más recomendable en este caso es cambiar el número de **CrimeId.** Esta solución también se hace con Python, en este caso, se crea un listado de los registros duplicados, luego con una sentencia for se van verificando uno por uno para ver si coincide con algún registro del dataframe, si este coincide, le cambio el número de CrimeId sumándole un numero 1, de esta forma eliminamos esos **CrimeId** duplicados.

Solución Nulos

Como se puede comprobar en la imagen anterior, los valores nulos están en las columnas City, State y en Range, esto no se tomo como un error, ya que, en el caso de los registros de ciudad nulos, si tenemos el State de este registro, por lo que por el estado podemos intuir la ciudad. En los registros vacíos de State, tenemos la dirección, por lo que intuimos que estado es, hay otros registros que tienen vacío State y City, pero si tenemos la dirección y si buscamos coincidencias, podemos determinar el State y City. Con el registro Range, no hay un dato exacto de que significa, por lo que no se toma como error.

Solución campos redundantes

La solución para estos campos redundantes, será eliminar los repetidos, en este caso, se decide eliminar la columna CallDateTime y quedarnos con la fecha (OffenseDate) y la hora (CallTime) en campos separados.

Para el campo Range se decide eliminar completamente ya que no aporta nada para un posible análisis.

Metodología

Para poder solucionar el problema de la limpieza del DataSet se siguió una serie de pasos los cuales se indican a continuación.

1. Instalar la librería Pandas en Python, para poder instalarla se puede hacer por medio de Pip con el siguiente comando: **pip install pandas.**
2. Reemplazar los delimitadores ";" por ", " para esto utilice la aplicación Visual Studio Code y su opción de buscar y reemplazar.
3. Importar la librería pandas en Python: `import pandas as pd`

4. Leer el archivo csv con pandas, para esto se puede utilizar la siguiente instrucción:
   `df = pd.read_csv('data_act_01.csv', engine= 'python')`
5. Se filtra el dataframe con los datos que mas interesan

```
df = df[['CrimeId','OriginalCrimeTypeName', 'OffenseDate', 'CallTime', 'Di
sposition', 'Address', 'City']]
```

6. Se crea una lista de los duplicados que se encuentren
   `duplicates = df[df['CrimeId'].duplicated(keep=False)]['CrimeId'].tolist()`
7. Se modifica el numero CrimeId de cada uno de los duplicados, esto se hizo porque se comprobó que lo único que se repetía era el numero de CrimeId y no los demás datos del registro.

```
set_duplicates = set(duplicates) # se crea un set de la lista de duplicado
s
for r in set_duplicates: #se recorre el set de duplicados y se compara con
 cada uno de los datos en el dataframe
    dup = df[df['CrimeId'] == r]
    df.loc[dup.iloc[1,:].name, 'CrimeId'] = df.loc[dup.iloc[1,:].name]['Cr
imeId'] + 1
```

8. Se verifica que efectivamente ya no haya ningún registro duplicado

```
df.duplicated().sum()
```

9. Se verifican los valores nulos.

```
df.isnull().sum() #Verificar nulos y contarlo
```

10. Por último, se debe valorar que información se necesita y que no, esta validación se hace verificando cada uno de los campos, si en estos hay repetidos, si existen valores nulos que no interesen, los campos que no aporten nada en una investigación se pueden obviar y eliminar.

Mejoras

1. Revisar el tipo de dato de cada uno de los campos, por ejemplo, si revisamos los tipos de datos de nuestro dataframe podemos verificar que solo CrimeId tiene tipo de datos int, los otros datos son tipo object.

```
df.dtypes
```

```
CrimeId                   int64
OriginalCrimeTypeName    object
OffenseDate              object
CallTime                 object
CallDateTime             object
Disposition              object
Address                  object
City                     object
State                    object
AgencyId                 object
Range                   float64
AddressType              object
dtype: object
```

   En este caso se recomienda cambiar el tipo de dato o castearlo para poder trabajar correctamente con el desde pandas, por ejemplo se puede hacer con el método **.astype()** en el cual indicamos entre los paréntesis el tipo de dato al que queremos castearlo.
2. Mayor granularidad en datos, por ejemplo para el campo de fecha u hora, se pueden separar por día, mes, año, por hora, minuto y segundo, esto para tener un mayor nivel de detalle en nuestro dataframe y poder trabajar mas fácilmente con las fechas, para extraer año, mes o día se puede realizar de la siguiente forma:

```
df.OffenseDate.dt.year # Extraer el año
df.OffenseDate.dt.month # Extraer el mes
df.OffenseDate.dt.day # Extraer el día
```

Para extraer la hora, minuto y segundo se puede realizar con la funciones **.dt.hour, .dt.minute() y .dt.second().**