

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA (FIME)

**MAESTRÍA EN CIENCIAS EN INGENIERÍA
DE SISTEMAS**

PORTAFOLIO DE EVIDENCIA

Ciencia de Datos

ALUMNO:
Evely Gutiérrez Noda

PROFESOR:
Dr. Elisa Schaeffer

Enero - Junio 2019

1. Introducción

El reporte muestra un resumen por prácticas realizadas a lo largo del semestre en la asignatura Ciencia de Datos. También se desarrolla una retroalimentación por parte del estudiante luego de la revisión de cada práctica por parte del profesor. la revisión de la tarea se presenta luego de cada retroalimentación.

2. Discución de la Práctica 1

La **Práctica 1** fue basada en la preparación de los datos con la herramienta Bash. Además se utilizó la herramienta **Gnuplot**, la cual se vinculó con **Python** para representar los grafos programados. El objetivo de la práctica es evitar hacer a mano preprocesamientos o limpiezas que se pueden hacer de forma automática, utilizar la herramienta Bash para evitar programar códigos largos y complicados para poder procesar los datos.

En el reporte de la práctica 1 se analiza el caso de estudio que permite evaluar las condiciones de reinserción social dentro del CERESO Apodaca y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres por separado, que se encuentran privados de libertad dentro del Centro de Reinserción. Se evalúan las condiciones de vida, económicas, educativas y familiares de los internos, así como las condiciones de los penitenciarios.

La tarea fue entregada en tiempo y calificada con calificación máxima 7 ya que no hubo errores.

E6 7)

REPORTE PRÁCTICA 1: Preparación de datos con bash

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda # 1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres por separado, que se encuentran privados de libertad dentro del Centro de Reinserción. Con este cuestionario se pretende evaluar las condiciones de vida, económicas, educativas y familiares de estas personas, así como las condiciones de los penitenciarios.

Se conforman diferentes preguntas con respecto a estos datos, que permitirán evaluar las condiciones de los internos en estos centros.

1. ¿El nivel de escolaridad tendrá valor significativo en la necesidad de cometer un delito?
Es decir, comprobar si el tener nivel escolar bajo pudiera afectar significativamente la necesidad de cometer un delito.
2. ¿Qué puede provocar que un interno no reciba visitas familiares?
3. ¿Se volverán adictos al cigarrillo u otra adicción los internos al entrar en los penitenciarios?
4. ¿En el caso de las mujeres, cual es el delito más frecuente por el cual son privadas de libertad?

Preparación de los datos

Se cuenta con dos encuestas, una aplicada a los hombres y otra a las mujeres, con alrededor de 287 preguntas cada una y una sesión de autoevaluación del estado de ánimo del encuestado. Las dos encuestas son muy parecidas, hacen diferencias en algunas preguntas referidas a las condiciones que tienen las mujeres en el penitenciario para tener a sus hijos, en caso que hayan dado a luz a su bebé estando bajo cumplimiento de condena, dado que viven con ellas sus primeros 2 años de vida.

Las preguntas de los cuestionarios giran sobre temas como: la infraestructura y los servicios de la penitenciaria, la situación familiar de los reclusos, si se realiza trabajo social con ellos, si los atienden clínicamente en caso de adicciones así como atención psicológica, sobre los servicios educativos y de capacitación, contiene temas de cultura y deporte, del ámbito laboral antes de ingresar al penitenciario y dentro del mismo, si cuentan o no con servicios de asesoría penal y jurídica, abordan el caso de la reinserción social de los reclusos y se tienen una sesión de preguntas referentes al estado de ánimo del recluso.

Se conoce que estas encuestas fueron respondidas por aproximadamente 312 hombres y 172 mujeres que se encuentran privados de libertad en CERESO en Apodaca. Fueron realizadas en papel y las respuestas se digitalizaron mediante formato **.CSV**. Estos datos se encuentran parcialmente limpios, por tanto, se procederá a revisar y limpiar nuevamente en caso de ser necesario.

Se utilizan herramientas de **Bash**, que es un lenguaje de consola para el manejo de distintos tipos de archivos. Se comienza haciendo copias de los archivos originales que se van a utilizar, en este caso, **backup-male.csv** para las respuestas a las encuestas de los hombres, **backup-female.csv** para las de mujeres y **backup-uniform.csv** para la unión de los datos de hombres y mujeres. Con estas copias se garantiza no dañar los datos originales en caso de llegar a cometer errores y así no perder información valiosa. Para hacer las copias de los archivos se usa el comando de bash **cp**.

```
cp uniform.csv backup-uniform.csv cp female.csv backup-female.csv cp  
male.csv backup-male.csv
```

Otra forma más abreviada de hacer las copias de los 3 archivos al mismo tiempo es:

11

```
for encuesta in "female" "male" "uniform"; do awk '{print "union" $0}'  
backup$encuesta.csv > backup$encuesta.csv; done
```

12345
13
Usando otra de las herramientas Bash **wc -l** se puede conocer la cantidad de filas en cada uno de los archivos de datos. Esta herramienta cuenta las líneas de los archivos, por tanto agrega una línea más correspondiente al encabezado de la columna, por tanto, la cantidad total de mujeres y hombres es uno menos de los totales que se muestran.

```
wc -l backup-* 173 backup-female.csv 313 backup-male.csv 485 backup-  
uniform.csv 971 total
```

Con esta instrucción se puede determinar la cantidad de renglones, palabras y caracteres de cada archivo, en este caso de todos los archivos copia.

```
wc backup-* 173 8546 258539 backup-female.csv 313 17467 487300  
backup-male.csv 485 18331 744894 backup-uniform.csv 971 44344 1490733  
total
```

Con otra de las herramientas Bash se pueden separar las columnas que se consideran útiles para dar respuesta a las preguntas planteadas en el inicio del reporte. Para esto se utiliza la herramienta **awk**, en este caso son las columnas 12: Nivel Escolar y 29: Primera vez

~~Interno. Esta información se obtiene del fichero backup-male.csv y se almacenan en otro fichero preguntalmale.csv por el comando de salida > .~~

```
awk -F',' '{print $12","$29}' backup-male.csv > preguntalmale.csv
```

Siguiendo con el uso del comando `awk` y las herramientas `sort` y `uniq` se puede ordenar las columnas deseadas y unificar por respuestas, para facilitar el análisis de los datos. En este ejemplo se trabaja con datos de los hombres y unifican por el nivel de escolaridad.

```
awk -F',' '{print $3 }' preguntalmale.csv | sort | uniq -c 1  
Comercial 1 escolLimpio 8 Licenciatura/profesional 1  
Licenciatura/profesional trunca 11 NA 12 Ningún estudio 24  
Preparatoria 6 Preparatoria trunca 71 Primaria 15 Primaria trunca 134  
Secundaria 24 Secundaria trunca 5 Técnico
```

Con estos resultados se puede ver que la mayoría de los internos hombres son de bajo nivel escolar, siendo el valor más alto 134 para la escolaridad de Secundaria y el segundo valor más alto es 71 correspondiente a Primaria.

Con la siguiente instrucción se puede revisar cuantos hombres estuvieron ya internos en el CERESO, arrojando que 247 fueron internos por primera vez, 60 fueron internos más de una vez y 5 no dieron respuesta a esta pregunta.

```
awk -F',' '{print $4}' preguntalmale.csv | sort | uniq -c 247 1 60 2  
5 NA
```

Se puede conocer si las familias de los internos se ven afectadas en el ambiente social (1: Si y 2: No), si aun así visitan a sus familiares internos (1: Si y 2: No) y la frecuencia con que lo hacen (1, 2, 3, 4 o 5 veces en la semana). En este ejemplo se puede conocer que, en algunos casos, a pesar de ser afectadas las familias en la sociedad por tener un familiar interno, aun así lo visitan de 1 a 2 veces en la semana. Esto se evidencia en los valores más altos (77, 72) de los siguientes resultados para el caso de los internos hombres.

```
awk -F',' '{print $2","$4","$3}' pregunta2male.csv | sort | uniq -c  
77 1,1,1 23 1,1,2 8 1,1,3 1 1,1,4 3 1,1,5 12 1,1,NA 72 1,2,1 27 1,2,2  
9 1,2,3 1 1,2,4 1 1,2,5 12 1,2,NA 16 1,NA,1 6 1,NA,2 1 1,NA,3 4  
1,NA,NA 18 2,1,NA 16 2,2,NA 4 2,NA,NA 1 NA,NA,NA
```

Para extraer de las mujeres algunos datos que combinen con los datos ya extraídos y analizados de los hombres, se trabaja sobre la pregunta 1, por tanto, se concatenaran los datos de las mujeres y los hombres en un mismo fichero con formato .CSV llamado **union.csv**. Se seleccionan las columnas 55 del archivo **backup-female.csv**. En el caso de las mujeres también se evidencia que son la mayoría de ellas de nivel bajo

mayores también se evidencia que son la mayoría de ellas de nivel bajo de escolaridad (73 Secundaria), pero se evidencia un número no muy bajo de mujeres con escolaridad universitaria (13 Licenciatura).

```
awk -F',' '{print $55}' backup-female.csv | sort | uniq -c 2  
Comercial 13 Licenciatura 4 Licenciatura trunca 2 NA 1 No tiene 28  
Preparatoria 1 Preparatoria técnica 7 Preparatoria trunca 27 Primaria  
1 Primaria trunca 1 Secretaria 73 Secundaria 4 Secundaria trunca 8  
Técnica
```

Del fichero **unifor.csv** donde se encuentran ya unificados los datos de los hombres y mujeres, se pueden obtener datos como la cantidad de internos en total que recibe visita o no.

```
awk -F',' '{print $189}' backup-uniform.csv | sort | uniq -c 1  
"familyVisits" 416 1 61 2 7 NA
```

Gracias al trabajo con estas herramientas bash se pudo limpiar un poco más los datos, ya que al unificar respuestas con el `uniq -c`, se descubrieron errores como por ejemplo: quedaban algunos "NO CONTESTO" y se cambiaron por NA, y respuestas hechas por personas a mano (ej. Tipo de homicidio).

This website does not host notebooks, it only renders notebooks available on other websites.

Delivered by Fastly, Rendered by Rackspace

nbviewer GitHub repository.

nbviewer version: aa567da

nbconvert version: 5.3.1

Rendered 2 minutes ago

1. Discución de la Práctica 2

La **Práctica 2** fue basada en la lectura y manipulación de los datos con la librería de Python llamada Pandas. El objetivo de la práctica es cargar los datos a un Data Frames utilizando la librería Pandas, ya que los datos están en formato .CSV y es una gran cantidad de información que al abrirlos en un Excel se perderían columnas.

Se presentan los ejemplos de trabajos con la librería Pandas, en este caso de estudio se pretende alcanzar objetivos como: leer, guardar y agregar columnas a un data frame, también se combinaron varios data frames y se realizaron filtros para practicar el trabajo con la librería Pandas.

La tarea fue entregada en tiempo y calificada con calificación 6 ya que se penalizó un punto por cambiar las respuestas con valor NAN a cero, error que da un valor a las respuestas vacías y afecta los resultados de la manipulación de datos.



REPORTE PRÁCTICA 2: Lectura y Manipulación de Datos con Pandas

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda # [1935050](#) (tel:1935050)

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres por separado, que se encuentran privados de libertad dentro del Centro de Reinserción. Con este cuestionario se pretende evaluar las condiciones de vida, económicas, educativas y familiares de estas personas, así como las condiciones de los penitenciarios.

El objetivo de este reporte es cargar los datos a un Data Frames utilizando la librería [Pandas](#) (<http://pandas.pydata.org>), ya que los datos están en formato **.CSV** y es una gran cantidad de información que al abrirlos en un Excel se perderían columnas. Se presentan los ejemplos de trabajos con la librería Pandas, en este caso de estudio se pretende alcanzar objetivos como:

1. Agregar columnas en común para hombres y mujeres en el data frame y guardarlas en un fichero **.CSV** nuevo usando la librería pandas. Algunas columnas fueron:
 - gender (correspondiente al género)
 - stayYears, stayMonths, stayDays (correspondiente al tiempo que llevan en el penitenciaro)
 - crime (correspondiente al crimen que cometieron)
 - sentenceYears, sentenceMonths, sentenceDays (correspondiente al tiempo de sentencia a partir del delito cometido)
2. Determinar qué cantidad de hombres y mujeres llevan privados de la libertad y aun están esperando sentencia.
3. Conocer el estado civil de los internos antes y después de entrar en el penal.
4. Indagar sobre que cantidad de hijos menores de edad tienen los internos y cuantos de ellos cuentan con seguro IMSS o Popular.
5. Trabajar sobre temas relacionados con los hijos que viven con las internas en el penal y los que no viven con ellas.

Además, se refleja el trabajo realizado con la librería Pandas en algunos ejemplos de:

- Leer data frame de archivos **.CSV**.
- Guardar el data frame en un fichero con formato **.CSV**.
- Agregar columnas de pruebas en el data frames.
- Combinar dos o más data frames (ej. el de hombres con el de mujeres).
- Realizar filtros a los renglones de un data frame para practicar el trabajo con la librería Pandas.

Para comenzar la lectura de datos y cargarlos en un Data Frame (ayuda para trabajar con [Data Frame](#) (https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html)), se necesita importar la librería Pandas, esto se logra con el siguiente código en la consola.

In []: `$ pip3 install pandas`

Ya teniendo instalado pandas, se comienza la lectura de datos con el siguiente código en **Python 3**. Primero se crea una clase llamada **ReadData** para la lectura del fichero **uniform.csv** usando la librería pandas mediante la variable `pd`. Luego se seleccionan las columnas que se van a utilizar y se guardan en un fichero nuevo llamado **UniformDataFrame.csv**, mediante la función de la librería pandas `.to_csv`.

A las columnas seleccionadas se le aplican diferentes correcciones, ya que pueden ser del tipo enteras o de caracteres, en este caso se reemplazaron los **NA** por ceros en columnas del tipo enteras y se cambiaron los **NA** por **No Contesto** en columnas de formato tipo caracteres. Además, se convirtieron a enteras las columnas que tratan con datos numéricos para posibles trabajos con ellos. Aun no quedan agregadas todas las columnas con las que se va a trabajar, pero es solo cuestión de agregarlas en el siguiente código.

```
In [12]: import pandas as pd
import numpy as np

class ReadData:

    def __init__(self):
        self.procesarUniform()

    def procesarUniform(self):
        uniform = pd.read_csv('uniform.csv')

        uniform['gender'] = uniform['gender']
        uniform['stayYears'] = uniform['stayYears'].replace(np.nan, 0)
        uniform['stayYears'] = uniform['stayYears'].astype(np.int64) #convierte a entero
        uniform['stayMonths'] = uniform['stayMonths'].replace(np.nan, 0)
        uniform['stayMonths'] = uniform['stayMonths'].astype(np.int64) #convierte a entero
        uniform['stayDays'] = uniform['stayDays'].replace(np.nan, 0)
        uniform['stayDays'] = uniform['stayDays'].astype(np.int64) #convierte a entero
        uniform['crime'] = uniform['crime'].replace(np.nan, 'No Contesto')
        uniform['sentenceYears'] = uniform['sentenceYears'].replace(np.nan, 0)
        uniform['sentenceYears'] = uniform['sentenceYears'].astype(np.int64) #convierte a entero
        uniform['sentenceMonths'] = uniform['sentenceMonths'].replace(np.nan, 0)
        uniform['sentenceMonths'] = uniform['sentenceMonths'].astype(np.int64) #convierte a entero
        uniform['sentenceDays'] = uniform['sentenceDays'].replace(np.nan, 0)
        uniform['sentenceDays'] = uniform['sentenceDays'].astype(np.int64) #convierte a entero
        uniform[['gender','stayYears','stayMonths','stayDays','crime','sentenceYears',
                 'sentenceMonths','sentenceDays']].to_csv('UniformDataFrame.csv')
```



Con esto se da cumplimiento al primer objetivo trazado en el inicio del reporte (Unificar datos de columnas seleccionadas de las encuestas de hombres y de mujeres en un fichero **.CSV** nuevo usando la librería pandas), el fichero **.CSV** nuevo que se crea es el **UniformDataFrame.csv**.

Luego se crea la clase **UniformMaleFemale** para inicializar los atributos que pertenecen a este nuevo fichero creado a partir de la lectura del Data Frame, los atributos corresponden a las columnas del fichero. Se programa un `if` para validar que los internos, ya sean hombres o mujeres, no lleven más años dentro del penitenciaro que la cantidad de años que les pusieron de sentencia por el delito cometido, es poco probable que esto pase, pero igual se valida. Se crean estas clases con el objetivo de mantener una organización que permita un mejor trabajo futuro.

```
In [13]: import numpy as np

class UniformMaleFemale:

    def __init__(self, folio, genero, crime):
        self.folio = folio
        self.gender = genero

        self.stayYears = 0
        self.stayMonths = 0
        self.stayDays = 0

        self.crime = crime
        self.sentenceYears = 0
        self.sentenceMonths = 0
        self.sentenceDays = 0

    if stayYears > sentenceYears:
        self.stayYears = 111111
    else:
        self.stayYears = stayYears
```

A continuación, se muestra el **main.py** con el cual se corre el programa, este tiene importadas las dos clases anteriormente descritas y únicamente ejecuta la inicialización de las variables en la clase **UniformMaleFemale** y guarda el DataFrame en la variable `rd`.

```
In [14]: from ReadData import ReadData
from UniformMaleFemale import UniformMaleFemale

print("leyendo Data Frame")
rd = ReadData()

leyendo Data Frame
```

Ya teniendo el archivo con los datos extraídos para analizar, se comienza a realizar algunas pruebas con ellos, por ejemplo, se busca la cantidad de hombres que responden detalladamente el tiempo que llevan dentro del penitenciario (respuestas con días, meses y años), de igual modo para las mujeres, y se busca también la cantidad de hombres y mujeres por independientes no responden a esa pregunta con tanta especificación. El código en Python para esto y los resultados obtenidos se muestran a continuación:

```
In [ ]: from ReadData import ReadData
from UniformMaleFemale import UniformMaleFemale

rd = ReadData()

def buscaCantidad(sexo, fichero, i, j, k):
    with open(fichero,'r') as uniform:
        todo = 0
        next(uniform)
        for lineas in uniform:
            temp = lineas.split(',')
            if str(temp[1]) == sexo:

                if int(temp[i]) > 0 and int(temp[j])>= 0 and int(temp[k])> 0:
                    todo = todo + 1
                    #print(temp)

            else:
                if int(temp[i]) > 0 and int(temp[j])> 0 and int(temp[k])> 0:
                    todo = todo + 1

    uniform.close()
    return todo

todoH = buscaCantidad('male', 'UniformDataFrame.csv', 2, 3, 4)
print('Hombres que responden detalladamente tiempo de estancia: ')
print(todoH)

todoM = buscaCantidad('female', 'UniformDataFrame.csv', 2, 3, 4)
print('Mujeres que responden detalladamente tiempo de estancia: ')
print(todoM)

todoH = buscaCantidad('male', 'UniformDataFrame.csv', 6, 7, 8)
print('Hombres que responden detalladamente tiempo de sentencia: ')
print(todoH)

todoM = buscaCantidad('female', 'UniformDataFrame.csv', 6, 7, 8)
print('Mujeres que responden detalladamente tiempo de sentencia: ')
print(todoM)
```

```
In [ ]: Hombres que responden detalladamente tiempo de estancia:
2
Mujeres que responden detalladamente tiempo de estancia:
0
Hombres que responden detalladamente tiempo de sentencia:
10
Mujeres que responden detalladamente tiempo de sentencia:
1
```

Se obtiene con este trabajo la información de que solo 2 hombres responden detalladamente al tiempo de estancia que llevan en el penitenciaro, esto puede ser utilizado para analizar el estado mental o psicológico de estos. Solo 10 dos hombres responden detalladamente el tiempo de sentencia impuesto, de un total de 296 Hombres y 144 Mujeres.

El trabajo anterior se pudiera resumir a dos líneas de código en Python solamente leyendo y filtrando el Data Frame y analizando el fichero resultante con herramientas de Bash. El siguiente ejemplo es para determinar el caso de los internos hombres cuántos de ellos responden detalladamente el tiempo de sentencia, primero se filtra el data frame por estas columnas, solo guardando las que sean mayores que cero, y luego se guardan los resultados en un fichero .CSV nuevo. En Bash se cuentan estos resultados y se sabe cuántos hombres responden detalladamente con día, mes y año el tiempo de sentencia que tienen. Es bueno hacer esta comparación para ver las utilidades y facilidades que brindan las herramientas **Bash** y el trabajo con Data Frame y la librería Pandas.

```
In [ ]: df6 = uniform[(uniform['gender'] == 'male') &(uniform['sentenceYears'] > 0) &(uniform['sentenceMonths'] > 0) &(uniform['sentenceDays'] > 0)]
df6[['sentenceYears', 'sentenceMonths', 'sentenceDays']].to_csv('TiempoSentenciaHombres.csv')
```

```
In [ ]: $ awk -F ',' '{print $2}' TiempoSentenciaHombres.csv | sort | uniq -c  
10 male
```

Otro trabajo realizado a estos datos es con respecto al objetivo 2, calcular cuántos hombres y mujeres están privados de la libertad y aun están esperando sentencia. Con el código siguiente en Python se calcula esto, arrojando que 68 hombres y 86 mujer están bajo estas condiciones.

```
In [ ]: def CantidadNoTiempoSentencia(sexo, fichero, i, j, k):  
    with open(fichero,'r') as uniform:  
        cantidad = 0  
        next(uniform)  
        for lineas in uniform:  
            temp = lineas.split(',')  
            if str(temp[1]) == sexo:  
                if int(temp[i]) == 0 and int(temp[j]) == 0 and int(temp[k]) == 0:  
                    cantidad = cantidad + 1  
                    #print(temp)  
    uniform.close()  
    return cantidad
```

```
In [ ]: Hombres sin sentencia:  
68  
Mujeres sin sentencia:  
86
```

En el caso del estado civil de las mujeres internas y los hombres antes y después de entrar en el interno. La pregunta correspondiente al estado Civil de los internos tiene 5 respuestas:

1. Soltera
2. Casada
3. Divorciada
4. Viuda
5. Unión Libre

Utilizando herramientas de Bash se llega a la información de que de las mujeres (47 Solteras, 46 Casadas, 13 Divorciadas, 13 Viudas y 55 Unión Libre) actualmente y (57 Solteras, 48 casadas, 9 Divorciadas, 7 Viudas y 49 Unión Libre) luego de entrar al penitenciarío. Por estos resultados se puede ver que el estado civil de las internas varió no muy significativamente. En el caso de los hombres si varió un poco más, sobre todo en el caso de los que tenían Unión libre antes de entrar al penitenciarío, poco más de la mitad varió, aunque también hubo una gran cantidad (207) que no contestaron a que estado civil tenían antes de entrar al penitenciarío. Con esta información se da por cumplido el objetivo 3 del reporte.

```
In [ ]: $ awk -F ',' '{print $2 "," $10}' UniformDataFrame.csv | sort | uniq -c # estado civil actual

47 female,1
46 female,2
13 female,3
13 female,4
53 female,5
2 male,0
93 male,1
90 male,2
30 male,3
8 male,4
89 male,5

$ awk -F ',' '{print $2 "," $11}' UniformDataFrame.csv | sort | uniq -c # estado civil antes de entrar al penitenciario
2 female,0
57 female,1
48 female,2
9 female,3
7 female,4
49 female,5
207 male,0
24 male,1
38 male,2
3 male,3
40 male,5
```

Para dar cumplimiento al objetivo 4 trazado en este reporte, se comienza por saber la cantidad de internos que tienen hijos (mujeres 155 Si y 17 No, hombres 238 Si y 67 No) lo cual informa que la mayoría de los internos tienen hijos.

```
In [ ]: $ awk -F ',' '{print $2 "," $12}' UniformDataFrame.csv | sort | uniq -c # Cantidad de hijos

155 female,1
17 female,2
7 male,0
238 male,1
67 male,2
```

Utilizando los ficheros nuevos creados a partir del filtrado del Data Frame se pudo conocer cuántos hijos de los internos tienen Seguro IMSS o Popular, en el caso de las mujeres de 155 que respondieron si tener hijos, 40 de ellos tienen seguro del IMSS y 68 tienen seguro Popular, quedando unas 47 internas que no dieron respuesta a esta pregunta.

```
In [4]: $ awk -F ',' '{print $2}' SeguroIMSS.csv | sort | uniq -c # Seguro IMSS
40 female
1 gender

$ awk -F ',' '{print $2}' SeguroPopular.csv | sort | uniq -c # Seguro Popular
68 female
1 gender

File "<tokenize>", line 3
17 female , 0
^

IndentationError: unindent does not match any outer indentation level
```

De igual modo en Python se realizó un trabajo de filtrado en el Data Frame que permitió almacenar la información referente a la cantidad de hijos de las internas que tienen seguro IMSS o Popular, cuantos hijos tienen y otros datos de interés en este estudio, y almacenarlos en ficheros por separados. También se concatenaron algunos data Frame usando el método de la librería Pandas .concat . Todo esto con el objetivo de ejercitarse el trabajo con la librería Pandas y tener la información más legible y organizada.

```
In [ ]: uniform = pd.read_csv('uniform.csv')

        df2 = uniform[(uniform['gender'] == 'female') & (uniform['hasChildren'] == 1) & (uniform['numberOfChildren'] > 0)]
        df2[['gender', 'hasChildren', 'numberOfChildren']].to_csv('cantidadhijos.csv')

        df3 = uniform[(uniform['gender'] == 'female') & (uniform['childrenSocialSecurityFOAQ'] == 1)]
        df3[['gender', 'childrenSocialSecurityFOAQ']].to_csv('seguroIMSS.csv')

        df4 = uniform[(uniform['gender'] == 'female') & (uniform['childrenSegPopFOAQ'] == 1)]
        df4[['gender', 'childrenSegPopFOAQ']].to_csv('seguroPopular.csv')

        df5 = pd.concat([df3, df4]).groupby(["childrenSocialSecurityFOAQ", "childrenSegPopFOAQ"], as_index=False)[["childrenSegPopFOAQ"]].sum()
        df5.to_csv('concatenados.csv')
```

En Python se realizó el cálculo de los hijos menores de edad y resolvió que 95 de 155 son menores de edad en el caso de las mujeres y 227 de 238 en el caso de los hombres, es decir, la mayoría son menores de edad.

```
In [ ]: #hijos menores de edad

def HijosMenoresDeEdad(sexo, fichero, i, j, k, l, m):
    with open(fichero, 'r') as uniform:
        cantidad = 0
        next(uniform)
        for lineas in uniform:
            temp = lineas.split(',')
            if str(temp[1]) == sexo:
                if int(temp[i]) < 16 and int(temp[j]) < 16 and int(temp[k]) < 16 and int(temp[l]) < 16 and int(temp[m]) < 16:
                    cantidad = cantidad + 1
                    #print(temp)
    uniform.close()
    return cantidad

HMDEM = HijosMenoresDeEdad('female', 'UniformDataFrame.csv', 13, 14, 15, 16, 17)
print('Cantidad de hijos menores de edad (mujeres internas): ')
print(HMDEM)

HMDHE = HijosMenoresDeEdad('male', 'UniformDataFrame.csv', 13, 14, 15, 16, 17)
print('Cantidad de hijos menores de edad (hombres internas): ')
print(HMDHE)
```

```
In [ ]: Cantidad de hijos menores de edad (mujeres internas):
95
Cantidad de hijos menores de edad (hombres internas):
227
```

Para el caso de las mujeres vale la pena hacer un estudio y conocer un poco más sobre los hijos que nacen en el penitenciario y viven con ellas hasta los 3 años de edad, dando cumplimiento al objetivo 5 trazado en el reporte. Por medio de herramientas **Bash** se obtuvo que 19 mujeres viven con su hijo en prisión y 17 de ellas dieron a luz allí.

```
In [ ]: $ awk -F ',' '{print $2 ", " $25}' UniformDataFrame.csv | sort | uniq -c # Cuantos hijos nacen en prisión
    1 gender,      howManyBornInPrisonFOAQ
148 female,      0
17 female,       1
7 female,        2

$ awk -F ',' '{print $2 ", " $26}' UniformDataFrame.csv | sort | uniq -c # Cuantos hijos viven en prisión
    1 gender,      childrenLiveInPrisonFOAQ
111 female,      0
19 female,       1
42 female,       2
```

Con respecto a las condiciones de cuidados y bienestar de los niños menores de 3 años que viven con las internas en la prisión, se conoce por análisis realizados a las respuestas de estas en las encuestas, que no son nada favorables para estos pequeños, dando como resultado que de 19 internas que si viven con sus hijos en el penal responden a que si la institución les da apoyo de la siguiente forma: 14 tienen guardería, 1 tienen zonas de juegos, 6 tienen juguetes, 9 tienen servicio médico, 5 les ponen vacunas a los niños, 7 tiene alimentación el menor y a 3 cuentan con ropa y zapatos. Tratándose de niños menores de 3 años deberían estar todos estos aspectos cubiertos al 100%, pero desgraciadamente no es así.

```
In [ ]: SAMSUNG@SAMSUNG-PC MINGW64 /e/DatosElisa
$ awk -F ',' '{print $2" , " $28}' UniformDataFrame.csv | sort | uniq -c
    150 female , 0.0
    1 gender , daycareFOAQ
14 female , 1
8 female , 2

    1 gender , playgroundFOAQ
150 female , 0
1 female , 1
21 female , 2

    1 gender , toysFOAQ
150 female , 0
6 female , 1
16 female , 2

    1 gender , childMedServFOAQ
150 female , 0
9 female , 1
13 female , 2

    1 gender , vaccinesFOAQ
150 female , 0
5 female , 1
17 female , 2

    1 gender , recFoodForChildrenFOAQCon
150 female , 0
7 female , 1
15 female , 2

    1 gender , childrensClothesFOAQ
150 female , 0
3 female , 1
19 female , 2
```

1. Discución de la Práctica 3

La **Práctica 3** consiste en realizar una estadística descriptiva de los datos. El objetivo de la práctica consiste en realizar conteos, promedios, correlaciones, junto con limpieza, categorización y cuantificación de la información.

La tarea fue entregada en tiempo y calificada con calificación máxima 7 ya que no hubo errores.

REPORTE PRÁCTICA 3: Estadística descriptiva Básica

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda # 1935050 (tel:1935050)

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

Se realizará una estadística descriptiva básica para obtener información de los datos. El objetivo de la práctica será realizar conteos, promedios, correlaciones, junto con limpieza, categorización y cuantificación de la información.

Desarrollo

De algunos de los ficheros guardados en la práctica anterior que ya están limpios y procesados para hacer ahora una estadística descriptiva básica de esta información. Con los ficheros correspondientes a los años de sentencia que tienen los hombres y las mujeres se puede conocer que en promedio de los años de sentencia de las mujeres es de 16.523255813953487 y el de los hombres de 14.94238683127572.

```
In [ ]: d = pd.read_csv("AnnosSentenciaMujeres.csv")
print(d.sentenceYears.describe())

count    86.000000
mean     16.523256
std      15.773937
min      0.000000
25%     7.000000
50%    11.500000
75%    24.750000
max     90.000000
Name: sentenceYears, dtype: float64

d = pd.read_csv("AnnosSentenciaHombres.csv")
print(d.sentenceYears.describe())

count    243.000000
mean     14.942387
std      12.731362
min      1.000000
25%     6.000000
50%    11.000000
75%    25.000000
max     92.000000
```

Se analizan también los delitos que cometen los internos, por los cuales están cumpliendo años de sentencia y en el nivel de educación de estos. Por ejemplo, las respuestas que dan los internos a la pregunta de qué delito cometieron, en el caso de cometer homicidio, responden de todas estas formas:

- Homicidio y otros delitos
- homicidio
- Homicidio
- Homicidio Accidental
- Homicidio calificado en grado de tentativa
- Homicidio calificado
- Homicidio Calificado
- Homicidio y secuestro
- Homicidio simple intencional
- Homicidio en riña
- Homicidio simple
- Homicidio Simple
- secuestro y homicidio
- Delitos contra la salud y homicidio
- Homicidio (tentativa)
- Homicidio imprudente

A continuación se evidencian los porcentajes de los delitos cometidos. Los promedios sacados antes de limpiar los datos no son agradables de revisar ya que la respuesta a el tipo de delito cometido por el interno es una respuesta a mano, es necesario limpiar la información y unificar los delitos más generales como: delitos de homicidios, de robos, de violaciones, de violencia familiar, de drogas, de salud, de posesión de armas, de secuestro u otros.

```
In [ ]: Robo  
10.23469387755102  
Homicidio  
22.035714285714285  
Violacion  
19.814814814814813  
Violencia Familiar  
7.25  
pension alimenticia  
1.0  
Drogas y Armas  
2.0  
Armas y Drogas  
11.0  
Contra la salud  
7.533333333333333  
nan  
nan  
Armas  
11.384615384615385  
Secuestro  
32.473684210526315  
lecciones y Robo  
3.0  
accidente  
6.0  
Chantaje  
9.0  
Federal  
12.5  
privacion de la libertad  
32.4  
Narcomenudeo  
8.0  
chantaje  
8.0  
Lesiones  
5.5  
delicuencia organizada  
8.5  
Empapelado  
14.0  
secuestro  
nan  
Drogas  
8.75  
dano a instituciones publicas  
8.0  
Contra la Salud  
3.0  
Fraude  
9.0  
Atentado al pudor  
nan  
contra la salud  
11.0  
Despojo de inmueble  
nan  
Trata de personas  
12.0  
Dano en propiedad ajena  
nan
```

Además se trabaja con el dropna() para no tomar en cuenta las respuestas donde no contestaron los internos, que se especifican con NAN.

La escolaridad de cada interno es un dato que pudiera ser importante para de algún modo concluir si afecta o no en que cometan delitos. Por ejemplo, se tiene que el nivel escolar más frecuente es Secundaria, lo cual pudiera demostrar que mientras menos nivel de educación tenga la persona es más propensa a cometer delitos.

```
In [ ]: print(data3.education.describe())
count          320
unique         17
top      Secundaria
freq          142
Name: education, dtype: object

## Cuantificando la cantidad de internos por nivel de escolaridad
data.education.unique()
    print(data.education.value_counts())

Cuantificando la educación
Secundaria           207
Primaria             98
Preparatoria        52
Secundaria truncada 28
Primaria truncada   16
Preparatoria truncada 13
Licenciatura          13
Ningún estudio       12
Técnica              8
Licenciatura/profesional 8
No contestó           6
Técnico               5
Licenciatura truncada 4
Comercial             3
Licenciatura/profesional truncada 1
Secretaria             1
Preparatoria técnica 1
No tiene               1

## Promedio de años de sentencia en cuanto a la escolaridad de cada interno
data3 = pd.read_csv('practica3b.csv')
for cri in data3.education.unique():
    print(cri)
    print(data3.loc[data3.education == cri]['sentenceYears'].mean())

Secundaria
14.43055555555555
Primaria
16.421875
Preparatoria
18.441176470588236
Ningún estudio
14.2
Técnico
11.0
Primaria truncada
18.454545454545453
Secundaria truncada
9.647058823529411
No contestó
12.25
Licenciatura/profesional
18.75
Preparatoria truncada
23.285714285714285
nan
nan
Licenciatura/profesional truncada
17.0
Comercial
```

20.0
Licenciatura
10.428571428571429
Técnica
16.16666666666668
Licenciatura trunca
12.0
Preparatoria técnica
25.0
Secretaria
5.0
No tiene
nan

Con respecto a los años de sentencia y al nivel de educación se puede ver que los internos con más años de sentencia se encuentran en preparatoria, ya sea terminada o no.

Estudiando los años de edad de los internos, se sacan los promedios de edades por los tipos de crímenes cometidos y se ve que las edades promedias oscilan entre los 15 y 60 años de edad. Además, la edad promedio de los internos es de 35.316770186335404 años. Se puede ver con esto que las personas de mayor edad, de 50 en adelante suelen cometer delitos de fraude, despojo de inmuebles o atentado al pudor y los más jóvenes cometan delitos de posesión de armas y drogas, o violencia familiar, robos, homicidios y otros.

```
In [ ]: # Promedio de edades
35.316770186335404
# Describiendo la edad de los internos
count    483.000000
mean     35.316770
std      12.584201
min      0.000000
25%     27.000000
50%     34.000000
75%     42.000000
max      99.000000
Name: age, dtype: float64
# Promedio de relación entre el delito cometido y la edad de los internos
Robo
30.970588235294116
Homicidio
36.379629629629626
Violacion
47.714285714285715
Violencia Familiar
33.33333333333336
pension alimenticia
35.0
Drogas y Armas
48.0
Armas y Drogas
23.0
Contra la salud
37.41860465116279
nan
nan
Armas
30.848484848484848
Secuestro
34.645833333333336
lecciones y Robo
42.0
accidente
33.0
Chantaje
26.0
Federal
33.66666666666664
privacion de la libertad
34.45454545454545
Narcomenudeo
```

```

38.0
chantaje
24.33333333333332
Lesiones
33.0
delicuencia organizada
36.0
Empapelado
15.0
secuestro
37.0
Drogas
35.83333333333336
dano a instituciones publicas
53.0
Contra la Salud
48.0
Fraude
50.2222222222222
Atentado al pudor
54.0
contra la salud
33.0
Despojo de inmueble
60.0
Trata de personas
46.8333333333336
Dano en propiedad ajena
45.0

```

Se va a intentar correlacionar las edades de los internos con los crímenes cometidos, el nivel de educación que tienen y los años de sentencia impuestos, con el objetivo de ver si existe relación entre estos aspectos y poder llegar a una idea de que factores pueden influir en que una persona cometiera un delito y la gravedad de este.

```

In [ ]: print(df.corr())

Matriz de correlación
      Unnamed: 0    crime  sentenceYears  education      age
Unnamed: 0    1.000000 -0.129235   0.059982 -0.117610  0.073101
crime        -0.129235  1.000000   0.156084  0.102415  0.014749
sentenceYears  0.059982  0.156084   1.000000 -0.070848  0.083818
education     -0.117610  0.102415   -0.070848  1.000000 -0.107225
age           0.073101  0.014749    0.083818 -0.107225  1.000000

```

```

In [ ]: Edad con años de sentencia
0.08381825523534246
Nivel escolar con delito cometido
0.10241540962615853
Nivel escolar con años de sentencia
-0.07084824936566585
Delito cometido con edad
0.014748906208745423

```

El coeficiente de correlación es una medida de correlación. Se calcula como un valor de punto flotante entre -1.0 y 1.0. Los resultados con valores de correlación cercanos a 1.0 serían los "correlacionados positivamente", lo que significa que tienden a moverse juntos. Los resultados con valores de correlación que se acercan a -1.0 se consideran "correlacionados negativamente" y generalmente se mueven en direcciones opuestas. Los valores de correlación cercanos a cero generalmente significan que los datos no están relacionados entre sí y se han movido independientemente en el período de tiempo reflejado en el cálculo de la correlación.

Se puede decir según los resultados anteriores que edad de los internos y los años de sentencia tienen una correlación positiva, lo cual podría decir que influye en la edad de la persona el delito que valla a cometer y por esto le darían más o menos años de sentencia, pero se necesitaría hacer pruebas de hipótesis para concluir esto.

Además, se puede plantear que el nivel escolar de la persona puede tener relación con el delito a cometer, ya que también tienen un coeficiente de correlación positivo.

Sin embargo, el nivel escolar va en dirección contraria con respecto a los años de sentencia, ya que es negativo el valor de correlación, lo cual se podría decir que a menor nivel escolar mayor cantidad de años de sentencia, lo cual comprueba que mientras más bajo sea la educación de la persona, peores y mayor cantidad de delitos puede cometer.

Con respecto al delito cometido y la edad de la persona también da un valor de correlación positivo, se pudiera decir que la edad condiciona de alguna manera el delito a cometer.

1. Discución de la Práctica 4

La **Práctica 4** consiste en visualizar la información utilizando Plotly. El objetivo de esta práctica es trabajar con Plotly para obtener cierto nivel de interacción con los datos ya que es una de las mejores librerías de uso libre para crear una variedad de gráficas interactivas y ricas en funcionalidad. Se crean gráficas de diferentes tipos como de caja-bigote, Histograma, Histograma horizontal, Histograma superpuesto y de barra.

La tarea fue entregada en tiempo y calificada con calificación máxima 7 ya que no hubo errores.



REPORTE PRÁCTICA 4: Visualización de información con plotly

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

El objetivo de esta práctica es trabajar con `plotly` para obtener cierto nivel de interacción con los datos. `Plotly` es una de las mejores librerías de uso libre para crear una variedad de gráficas interactivas y ricas en funcionalidad. Se van a crear algunas graficas de diferentes tipos que permitirán concluir algunas cosas sobre los datos que se obtuvieron de las encuestas. Se incluyen gráficas de caja-bigote, histograma, histograma horizontal, histograma superpuesto y de barra.

`Plotly` ofrece un API excelente para crear gráficas interactivas que pueden ser incluidas en webs y blogs. El paquete no viene incluido, pero se puede instalar de manera muy sencilla con pip:

```
$pip install plotly
```

Para poder utilizar `plotly` se necesitan credenciales, por lo que es necesario registrarse. Este registro es gratuito, sin embargo, el número de archivos privados que se pueden almacenar es limitado en la versión gratuita.

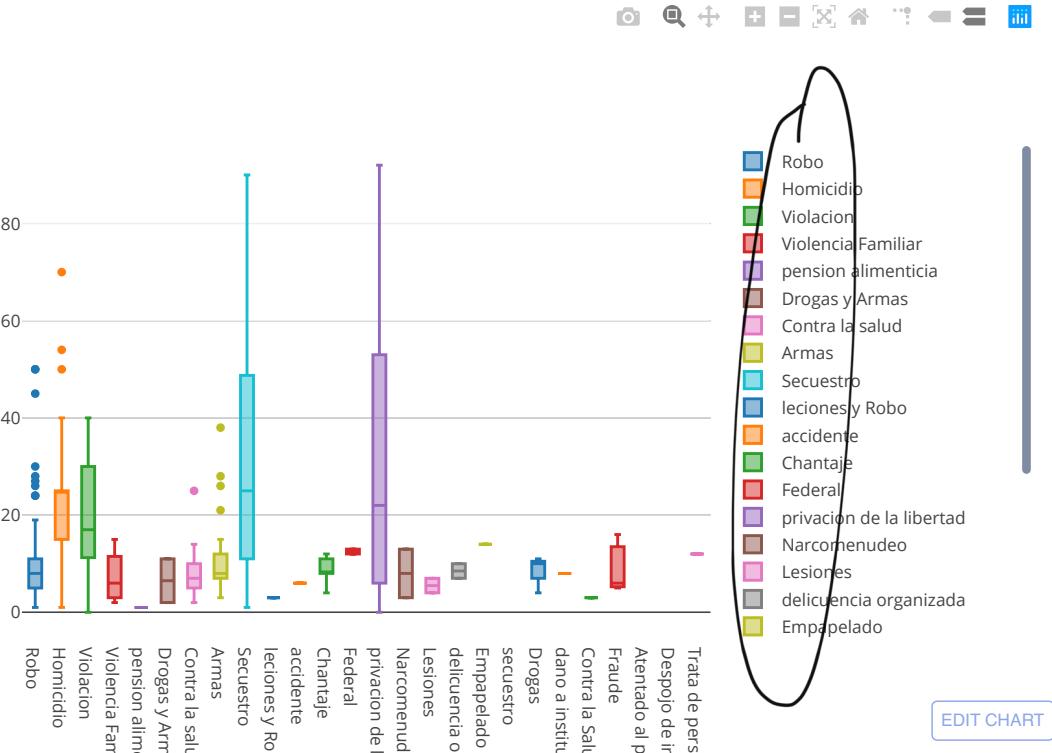
Luego de tener ya lista la librería, se leen los datos guardados en la práctica anterior [Práctica 3](#) (<https://github.com/EvelyGutierrez/Analisis-Estadistico-Multivariado/blob/master/ReportePractica3Evely.ipynb>), se importan las funcionalidades a utilizar y se cambian las credenciales por el APY KEY que se genera luego de crear un usuario en `plotly`. Luego se crea la gráfica de caja-bigote con el código siguiente que se puede correr en [jupyter](#), dando como resultado el gráfico siguiente.

```
In [171]: import plotly as plotly
import plotly.plotly as py
import plotly.graph_objs as go
import pandas as pd
import ssl

plotly.tools.set_credentials_file(username='evely', api_key='2e7wTIIQwgd8kwBZbDOE')

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")
datos = []
for sem in d.crime.unique():
    if sem != '':
        datos.append(go.Box(y = d.loc[d['crime'] == sem].sentenceYears, name = sem))
py.iplot(datos, filename='Crimen_Años de sentencia')
```

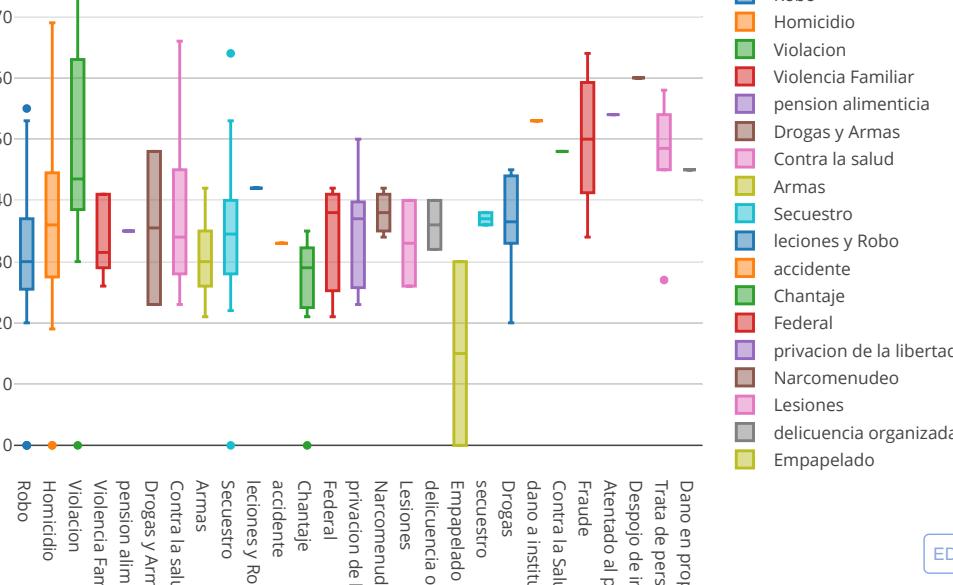
Out[171]:



```
In [173]: import pandas as pd
from numpy import NaN
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")
datos = []
for sem in d.crime.unique():
    if sem != '':
        datos.append(go.Box(y = d.loc[d['crime'] == sem].age, name = sem))
py.iplot(datos, filename='Crimen Edad')
```

Out[173]:

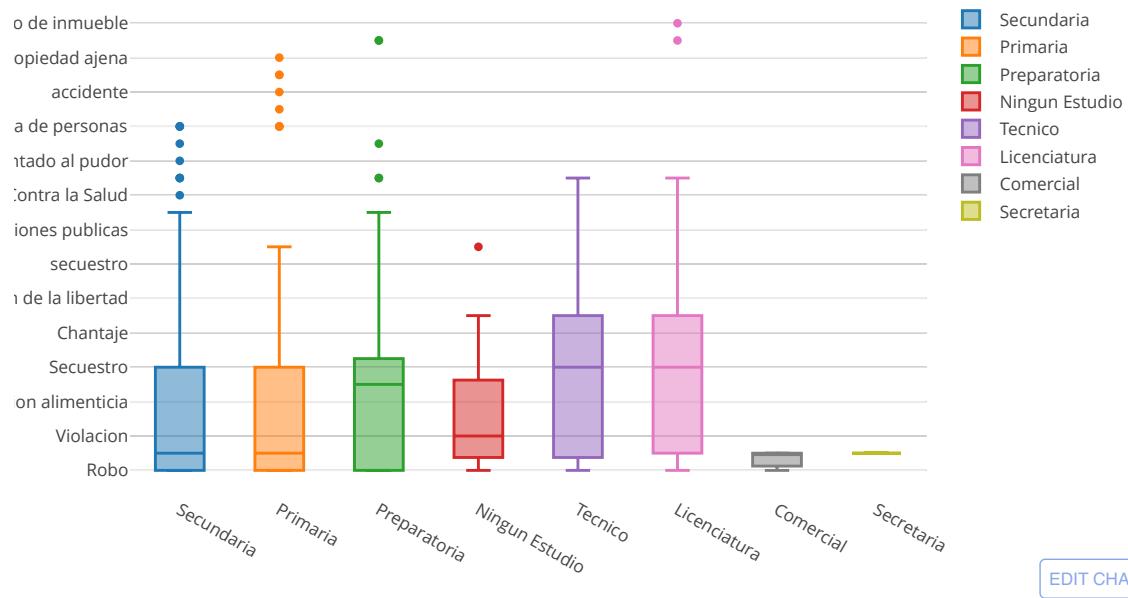


Otro gráfico que se realizó fue el de nivel educacional con respecto al crimen cometido, en este caso se limpian un poco más los datos de nivel de educación. Luego se realizó otro gráfico con el nivel de educación y los años de sentencia de los internos. Se puede ver por ejemplo que los internos que respondieron a "ningún estudio" cometen el delito contra la salud más frecuentemente y la media de casi todos los internos cometen homicidios, independientemente del nivel educacional que tengan.

```
In [174]: import pandas as pd
from numpy import NaN
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")
datos = []
for sem in d.education.unique():
    if sem != '':
        datos.append(go.Box(y = d.loc[d['education'] == sem].crime, name = sem))
py.iplot(datos, filename='Educacion_Crimen')
```

Out[174]:



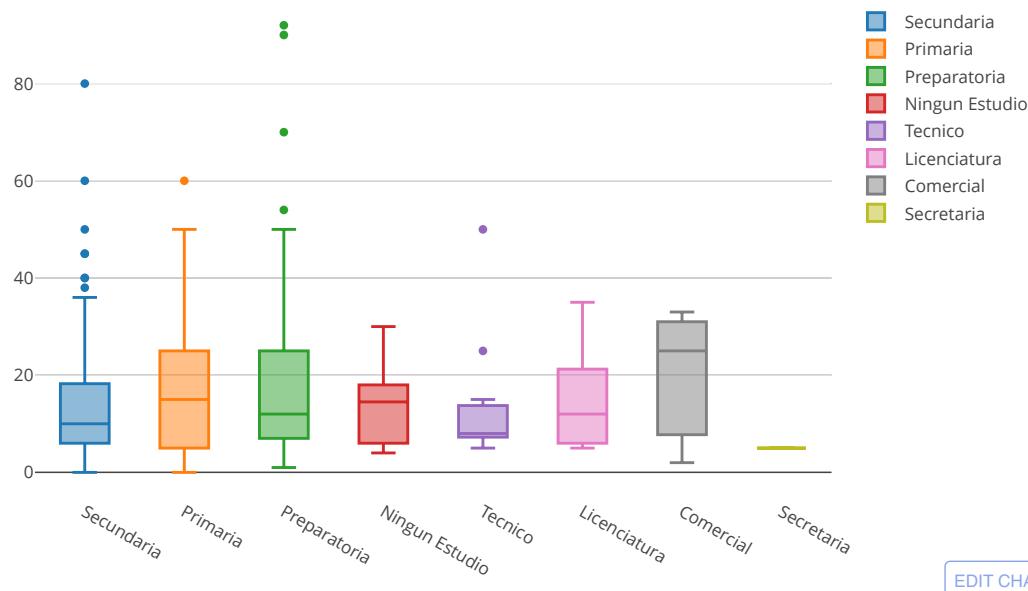
[EDIT CHART](#)

Se graficó el nivel de educación con respecto a los años de sentencia impuestos, donde se evidencia que las mayores sentencias impuestas son a los internos con escolaridad Preparatoria.

```
In [175]: import pandas as pd
from numpy import NaN
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")
datos = []
for sem in d.education.unique():
    if sem != '':
        datos.append(go.Box(y = d.loc[d['education'] == sem].sentenceYears, name = sem))
py.iplot(datos, filename='Educacion_Sentencia')
```

Out[175]:



[EDIT CHART](#)

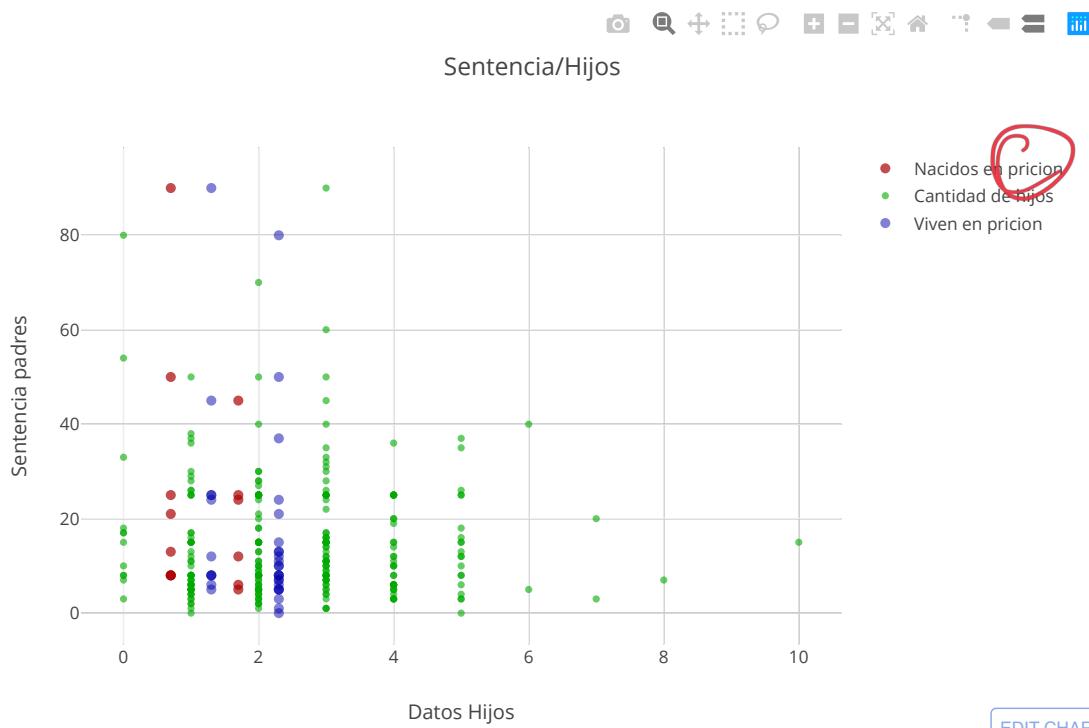
El siguiente gráfico es con relación a la cantidad de hijos que tienen los internos y a los años de sentencia impuestos, de modo que se puede apreciar que si tiene un número de años alto y más de dos hijos, pudiera ser que el crimen que cometan sea de categoría más grave, teniendo así mayor cantidad de años de sentencia.

```
In [176]: import pandas as pd
import ssl
import plotly.plotly as py
import plotly.graph_objs as go
from random import randint

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")
m1 = {'size': 7, 'color': 'rgba(170, 0, 0, .7)'}
m2 = {'size': 5, 'color': 'rgba(0, 170, 0, .6)'}
m3 = {'size': 7, 'color': 'rgba(0, 0, 170, .5)'}

delta = 0.3
datos = [go.Scatter(x = d.howManyBornInPrisonFOAQ - delta, y = d.sentenceYears, mode = 'markers', marker = m1, name="Nacidos en prisión"),
         go.Scatter(x = d.numberOfChildren, y = d.sentenceYears, mode = 'markers', marker = m2, name="Cantidad de hijos"),
         go.Scatter(x = d.childrenLiveInPrisonFOAQ + delta, y = d.sentenceYears, mode = 'markers', marker = m3, name="Viven en prisión")]
f = {"data": datos,
      "layout" : {
          "title": "Sentencia/Hijos",
          "showlegend": True,
          "xaxis": {
              "title": "Datos Hijos",
              "zeroline": False,
          },
          "yaxis": {
              "title": "Sentencia padres",
              "zeroline": False,
          }
      }
}
py.iplot(f, filename='Sentencias_Hijos')
```

Out[176]:



Otros ejemplos de diagramas son el Histograma, Histograma superpuesto y de barra, los cuales evidencian la misma información mostrada en los diagramas anteriores pero en diferentes tipos de diagramas.

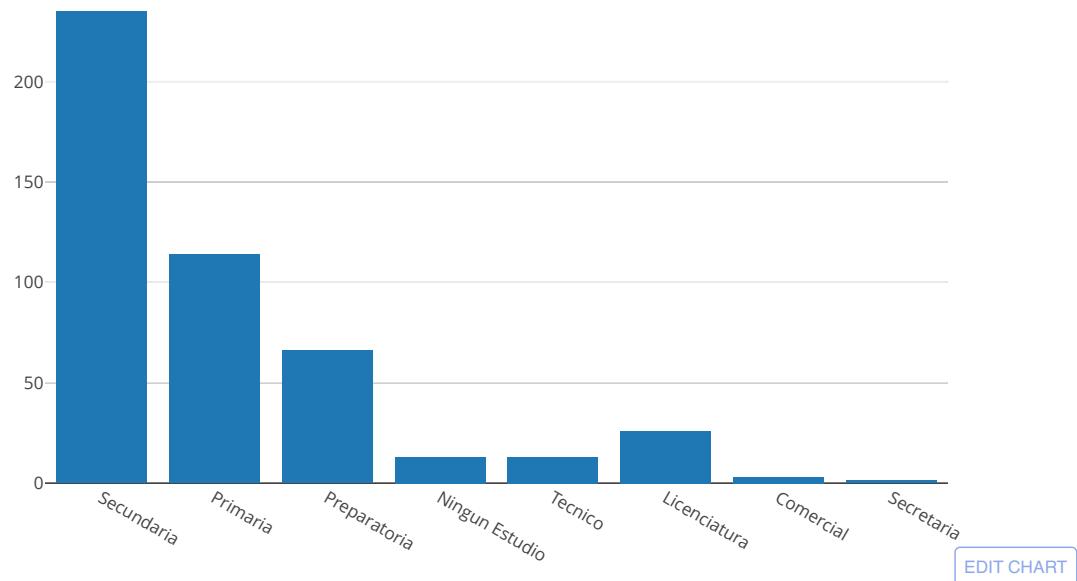
```
In [184]: import plotly.plotly as py
import plotly.graph_objs as go

df = pd.read_csv("practica3ok.csv")

data = [go.Histogram(x=df.education,
                     y=df.sentenceYears)]

py.iplot(data, filename='Educación_Sentencia')
```

Out[184]:



[EDIT CHART](#)

```
In [185]: import plotly.plotly as py
import plotly.graph_objs as go

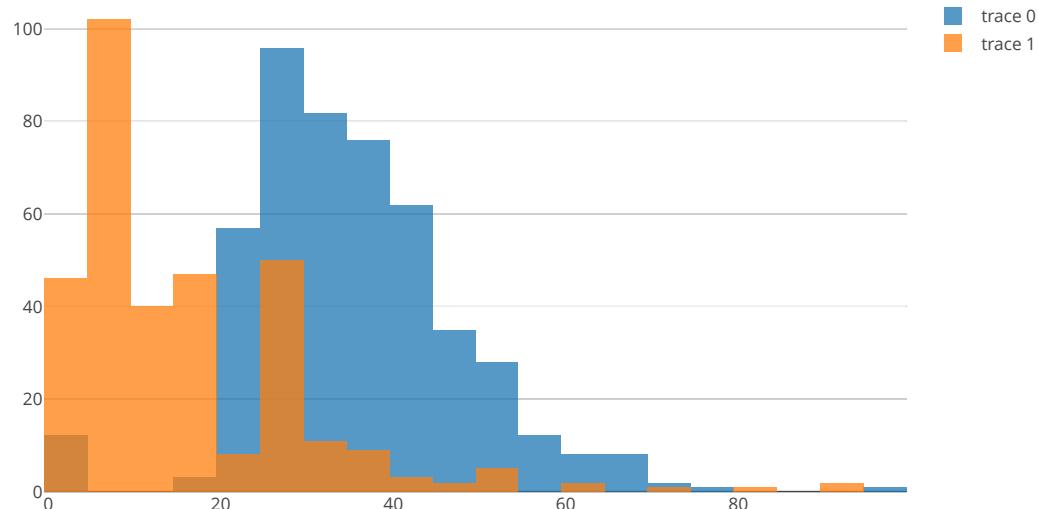
df = pd.read_csv("practica3ok.csv")
edad = df.age
sentencia = df.sentenceYears

trace1 = go.Histogram(
    x=edad,
    opacity=0.75
)
trace2 = go.Histogram(
    x=sentencia,
    opacity=0.75
)

data = [trace1, trace2]
layout = go.Layout(barmode='overlay')
fig = go.Figure(data=data, layout=layout)

py.iplot(fig, filename='Histograma_Superpuesto')
```

Out[185]:



[EDIT CHART](#)

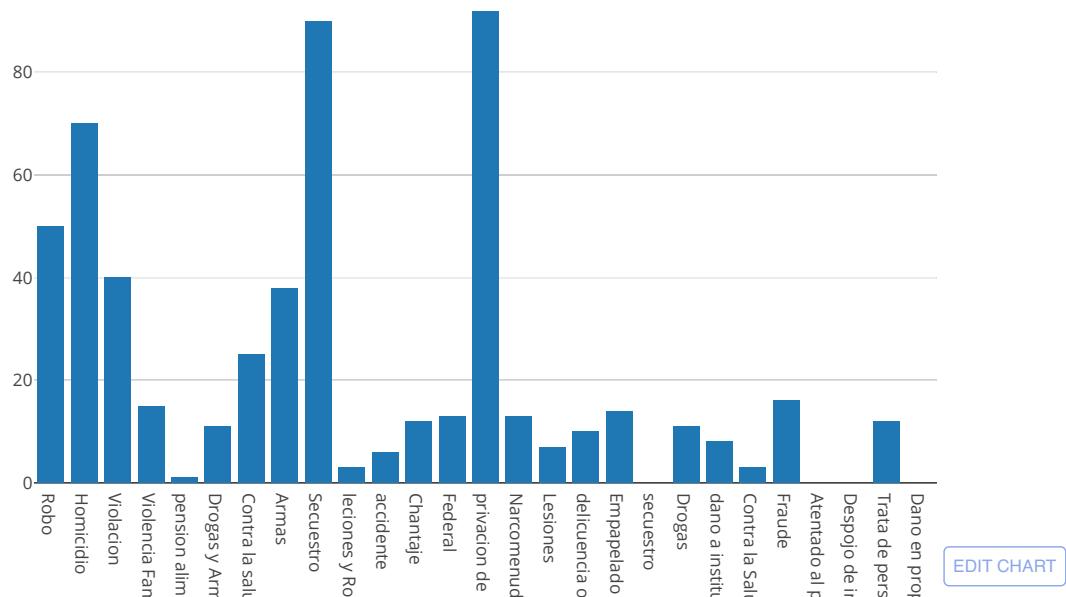
```
In [183]: import plotly.plotly as py
import plotly.graph_objs as go

df = pd.read_csv("practica3ok.csv")

data = [go.Bar(x=df.crime,
                y=df.sentenceYears)]

py.iplot(data, filename='Crime_Sentencia')
```

Out[183]:

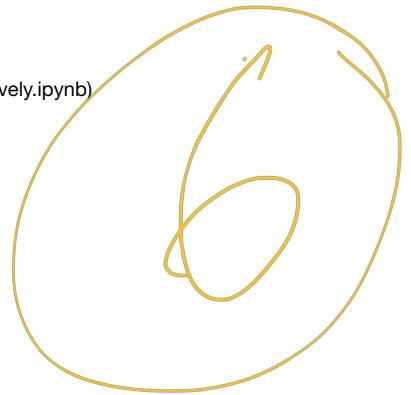


El uso de los diagramas facilita la interpretación de la información, la relaciona y permite llegar a conclusiones, en el caso de los datos que se están analizando, por medio de los diagramas también se pudo ver la relación existente entre las columnas seleccionadas para el análisis de esta información.

1. Discución de la Práctica 5

La **Práctica 5** consiste en realizar pruebas estadísticas a los datos. En la práctica anterior se mostraron varias gráficas que permitieron asumir algunas conclusiones a partir de la información capturada de las encuestas. El objetivo de esta práctica consiste en realizar pruebas de hipótesis para poder comprobar o no lo que se asume en la práctica anterior. Las pruebas de hipótesis realizadas fueron Shapiro-Wilk, Kruskal-Wallis y Friedman Test. Se utilizaron diagramas como cuantiles-cuantiles e histogramas de barras para visualizar los resultados.

La tarea fue entregada en tiempo y calificada con calificación 6 ya que se penalizó un punto por errores ortográficos y errores en la representación de algunos datos que no se realizó una buena limpieza antes de visualizarlos.



REPORTE PRÁCTICA 5: Pruebas estadísticas

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción. La práctica anterior mostró varias gráficas que permitieron asumir algunas conclusiones a partir de la información capturada de las encuestas. El objetivo de esta práctica es realizar pruebas de hipótesis para poder comprobar o no lo asumido en la práctica anterior.

Desarrollo

Se presume que existe alguna relación entre la edad que tienen los internos y el tipo de crimen que cometieron, así mismo sucede entre el nivel de educación que tienen los internos con el crimen cometido y por ende con los años de sentencia impuestos. En el siguiente reporte se verifica esta relación por medio de pruebas estadísticas, y se analiza los hombres y mujeres por separados. Primeramente se recuerda que la media de las edades de los internos hombres y mujeres tienen una media entre los 35 y 36 años, luego se agreguemos una línea vertical para marcar 35 para marcar esta media. Se realizó lo mismo para el caso de los años de sentencia dependiendo el género, donde la media es 14 para los hombres y 16 para las mujeres. El histograma es un gráfico simple y comúnmente usado para verificar rápidamente la distribución de una muestra de datos [Normality Assumption](https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/) (<https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/>).

```
In [6]: import pandas as pd

data3 = pd.read_csv('practica3ok.csv')
for cri in data3.gender.unique():
    print('media edad por genero')
    print(cri)
    print(data3.loc[data3.gender == cri]['age'].mean())

    print('media sentencia por genero')
    print(cri)
    print(data3.loc[data3.gender == cri]['sentenceYears'].mean())
```

```
media edad por genero
male
35.092948717948715
media sentencia por genero
male
14.94238683127572
media edad por genero
female
35.72514619883041
media sentencia por genero
female
16.523255813953487
```

Histograma de las edades de los internos hombres

```
In [1]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib as plt
from numpy.random import randn
from numpy import concatenate
import plotly.graph_objs as go
import plotly.plotly as py

d = pd.read_csv("practica3ok.csv")

m = d.loc[d.gender == 'male'].age

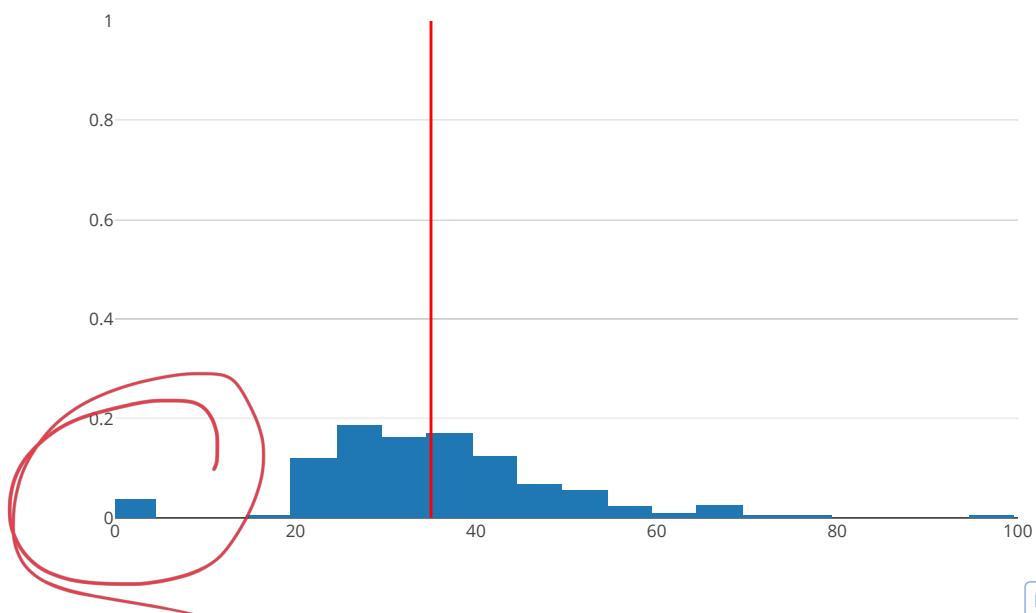
layout = {'xaxis': {'range': [0, 100]}, 'yaxis': {'range': [0, 1]}, \
'shapes': [{ 'type': 'line', 'x0': 35, 'y0': 0, 'x1': 35, 'y1': 1, \
'line': { 'color': 'rgb(255, 0, 0)', 'width': 2}}]}

py.iplot({'data': [go.Histogram(x = m, histnorm='probability')], 'layout': layout}, filename='male_e
dad')
```

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\display.py:689: UserWarning:

Consider using IPython.display.IFrame instead

Out[1]:



Histograma de las edades de las internas mujeres

```
In [2]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib as plt
from numpy.random import randn
from numpy import concatenate
import plotly.graph_objs as go
import plotly.plotly as py

d = pd.read_csv("practica3ok.csv")
fe = d.loc[d.gender == 'female'].age

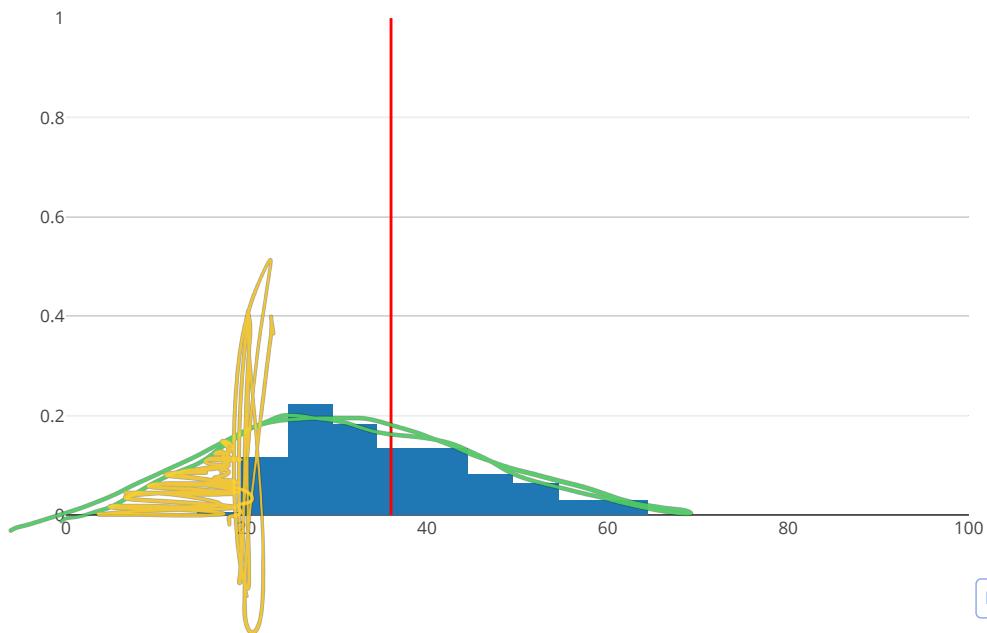
layout = {'xaxis': {'range': [0, 100]}, 'yaxis': {'range': [0, 1]}, \
'shapes': [{ 'type': 'line', 'x0': 36, 'y0': 0, 'x1': 36, 'y1': 1, \
'line': { 'color': 'rgb(255, 0, 0)', 'width': 2}}]}

py.iplot({'data': [go.Histogram(x = fe, histnorm='probability')], 'layout': layout}, filename='femal e_edad')
```

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\display.py:689: UserWarning:

Consider using IPython.display.IFrame instead

Out[2]:



[EDIT CHART](#)

Histograma de la sentencia de los hombres

```
In [3]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib as plt
from numpy.random import randn
from numpy import concatenate
import plotly.graph_objs as go
import plotly.plotly as py

d = pd.read_csv("practica3ok.csv")

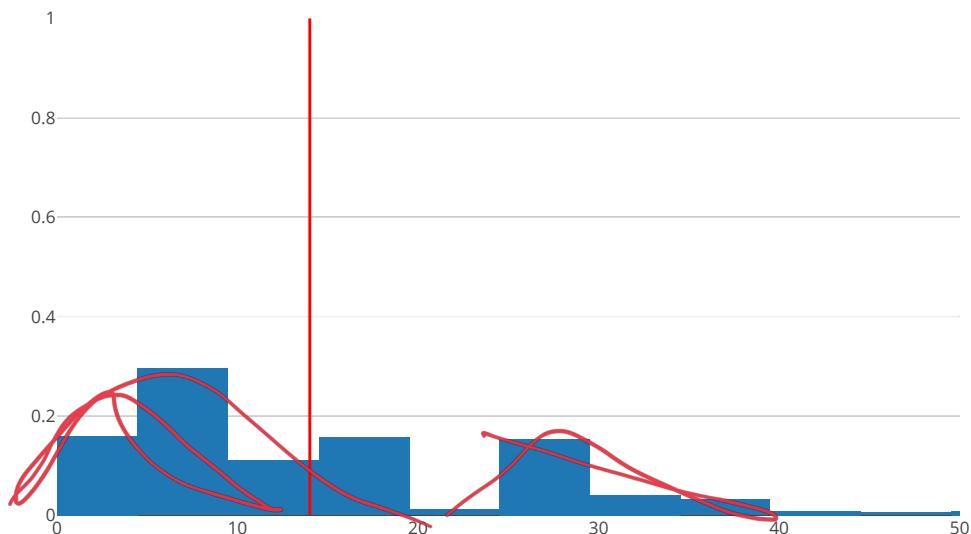
m = d.loc[d.gender == 'male'].sentenceYears

layout = {'xaxis': {'range': [0, 50]}, 'yaxis': {'range': [0, 1]}, \
'shapes': [{ 'type': 'line', 'x0': 14, 'y0': 0, 'x1': 14, 'y1': 1, \
'line': {'color': 'rgb(255, 0, 0)', 'width': 2}}]}

py.iplot({'data': [go.Histogram(x = m, histnorm='probability')], 'layout': layout}, filename='male_s
entencia')
```

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\display.py:689: UserWarning:
Consider using IPython.display.IFrame instead

Out[3]:



[EDIT CHART](#)

Histograma de la sentencia de las mujeres

```
In [4]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib as plt
from numpy.random import randn
from numpy import concatenate
import plotly.graph_objs as go
import plotly.plotly as py

d = pd.read_csv("practica3ok.csv")

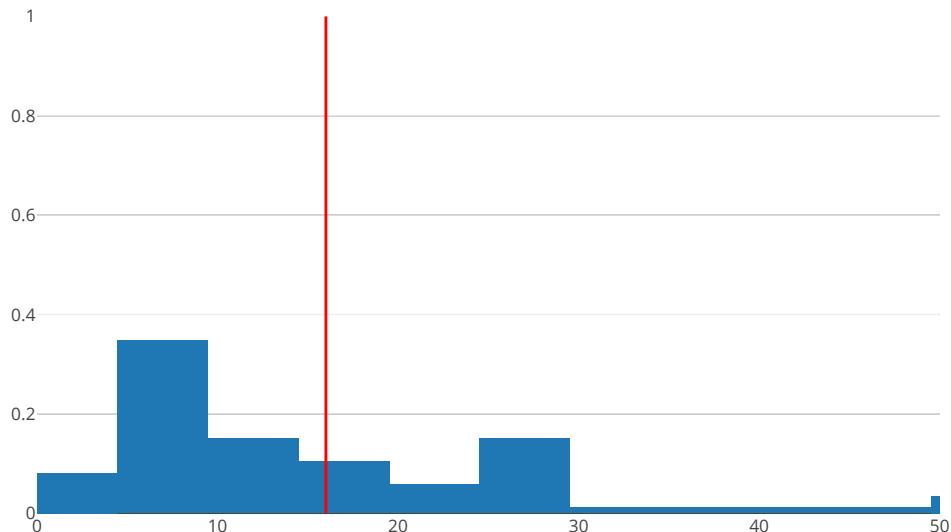
m = d.loc[d.gender == 'female'].sentenceYears

layout = {'xaxis': {'range': [0, 50]}, 'yaxis': {'range': [0, 1]}, \
'shapes': [{ 'type': 'line', 'x0': 16, 'y0': 0, 'x1': 16, 'y1': 1, \
'line': { 'color': 'rgb(255, 0, 0)', 'width': 2}}]}

py.iplot({'data': [go.Histogram(x = m, histnorm='probability')], 'layout': layout}, filename='female_sentence')
```

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\display.py:689: UserWarning:
Consider using IPython.display.IFrame instead

Out[4]:



[EDIT CHART](#)

Los histogramas anteriores se usan para tener una idea de si los datos utilizados pudieran ser normales o si se ven bimodales. Se puede apreciar que la distribución no es muy parecida a la gaussiana, se puede ver una forma similar a la de Gauss cuando el histograma tiene una forma aproximada a una campana. Se calcula la desviación estandar de la muestra de los datos a analizar con el código siguiente, para poder realizar la prueba de normalidad a estos datos. Se utiliza la función `randn()` de la librería **NumPy** para generar números gaussianos aleatorios con una media y una desviación estándar que se obtiene de los datos a analizar que se describen en el siguiente código. Estos nuevos datos generados se les llama datos auxiliares para comparar visualmente cuando los datos pueden ser normalmente distribuidos, unimodales y bimodales. Se agrega además una línea vertical para marcar 35 que es la media de las edades de los internos, y luego la linea roja estará en 15 que es la media de los años de sentencia de los internos, al igual que en los histogramas anteriores.

```
In [5]: import pandas as pd

data3 = pd.read_csv('practica3ok.csv')
for cri in data3.gender.unique():
    print(cri)
    print('desviacion estandar edad')
    print(data3.loc[data3.gender == cri]['age'].std())

    print(cri)
    print('desviacion estandar sentencia')
    print(data3.loc[data3.gender == cri]['sentenceYears'].std())

male
desviacion estandar edad
13.649529688984915
male
desviacion estandar sentencia
12.731361910560755
female
desviacion estandar edad
10.387798801916043
female
desviacion estandar sentencia
15.77393736082406
```

```
In [42]: import plotly.plotly as py
import plotly.graph_objs as go
import pandas as pd
import ssl
import plotly.plotly as py

d = pd.read_csv("practica3ok.csv")

m = d.loc[d.gender == 'male'].sentenceYears

layout = {'xaxis': {'range': [0, 50]}, 'yaxis': {'range': [0, 1]}, \
'shapes': [{'type': 'line', 'x0': 14, 'y0': 0, 'x1': 14, 'y1': 1, \
'line': {'color': 'rgb(255, 0, 0)', 'width': 2}}]}

from numpy.random import randn
from numpy.random import seed
from numpy import concatenate

seed(7)
unimodal = 13 * randn(1000) + 14 # varianza y media
bimodal = concatenate((unimodal, 6 * randn(1000) + 10))
print((py.iplot({'data': [go.Histogram(x = unimodal, histnorm='probability')], 'layout': layout}, \
filename="unimodal_male_sentencia")).embed_code)
print((py.iplot({'data': [go.Histogram(x = bimodal, histnorm='probability')], 'layout': layout}, \
filename="bimodal_male_sentencia")).embed_code)

<iframe id="igraph" scrolling="no" style="border:none;" seamless="seamless" src="https://plot.ly/~evely/150.embed" height="525px" width="100%">></iframe>

C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\display.py:689: UserWarning:
Consider using IPython.display.IFrame instead
```

Histograma Unimodal sentencia hombres (datos auxiliares)

unimodal_male_sentencia.png

Histograma Bimodal sentencia hombres (datos auxiliares)

Todos los diagramas mostrados y los referentes a los datos de sentencias de hombres y mujeres están disponibles en [esta página](https://github.com/EvelyGutierrez/Analisis-Estadistico-Multivariado/tree/master/DiagramasPractica5) (<https://github.com/EvelyGutierrez/Analisis-Estadistico-Multivariado/tree/master/DiagramasPractica5>).

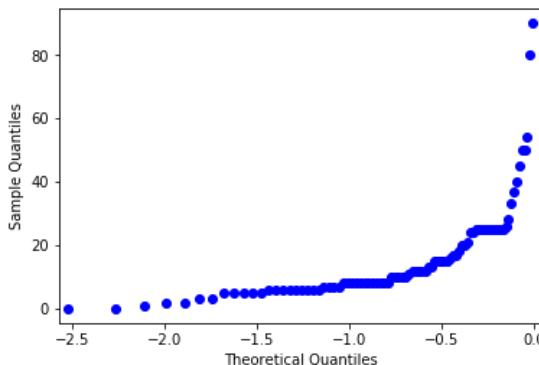
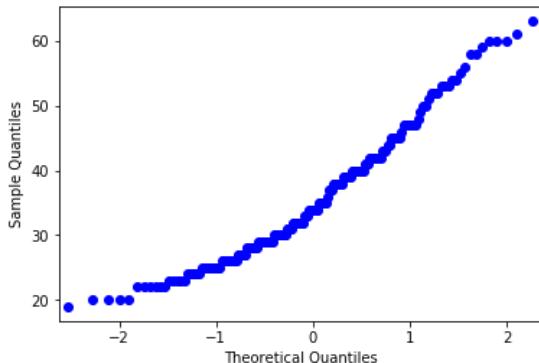
Se observa una forma similar a la de Gauss en los datos graficados auxiliares, que aunque no es fuertemente la forma de la campana, es una aproximación.

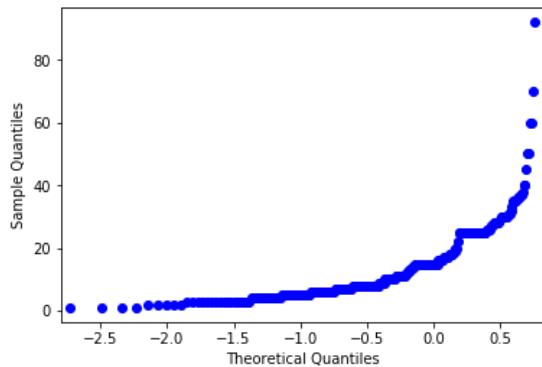
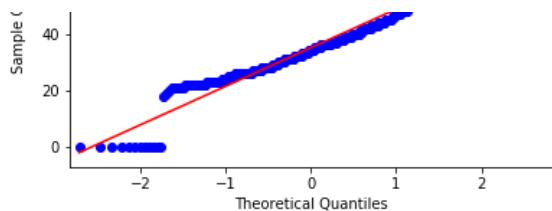
Otra gráfica que permite verificar la distribución de una muestra de datos es la gráfica de **cuantiles-cuantiles** o gráfica QQ para abreviar. Estas gráficas generan su propia muestra de la distribución idealizada con la que se está comparando, en este caso la distribución gaussiana. Estos gráficos QQ se realizarán con el siguiente código usando la función `qqplot()`. Esta función toma la muestra de datos y la compara con una distribución gaussiana. Se dibuja una línea roja estandarizada que establece la presencia de distribución gaussiana.

```
In [50]: import pandas as pd
from statsmodels.graphics.gofplots import qqplot
import matplotlib as plt

d = pd.read_csv("practica3ok.csv")
m = d.loc[d.gender == 'female'].age # mujeres edad
m1 = d.loc[d.gender == 'female'].sentenceYears # mujeres sentencia
h = d.loc[d.gender == 'male'].age # hombres edad
h1 = d.loc[d.gender == 'male'].sentenceYears # hombres sentencia

f = qqplot(m, line='s')
f = qqplot(m1, line='s')
f = qqplot(h, line='s')
f = qqplot(h1, line='s')
```

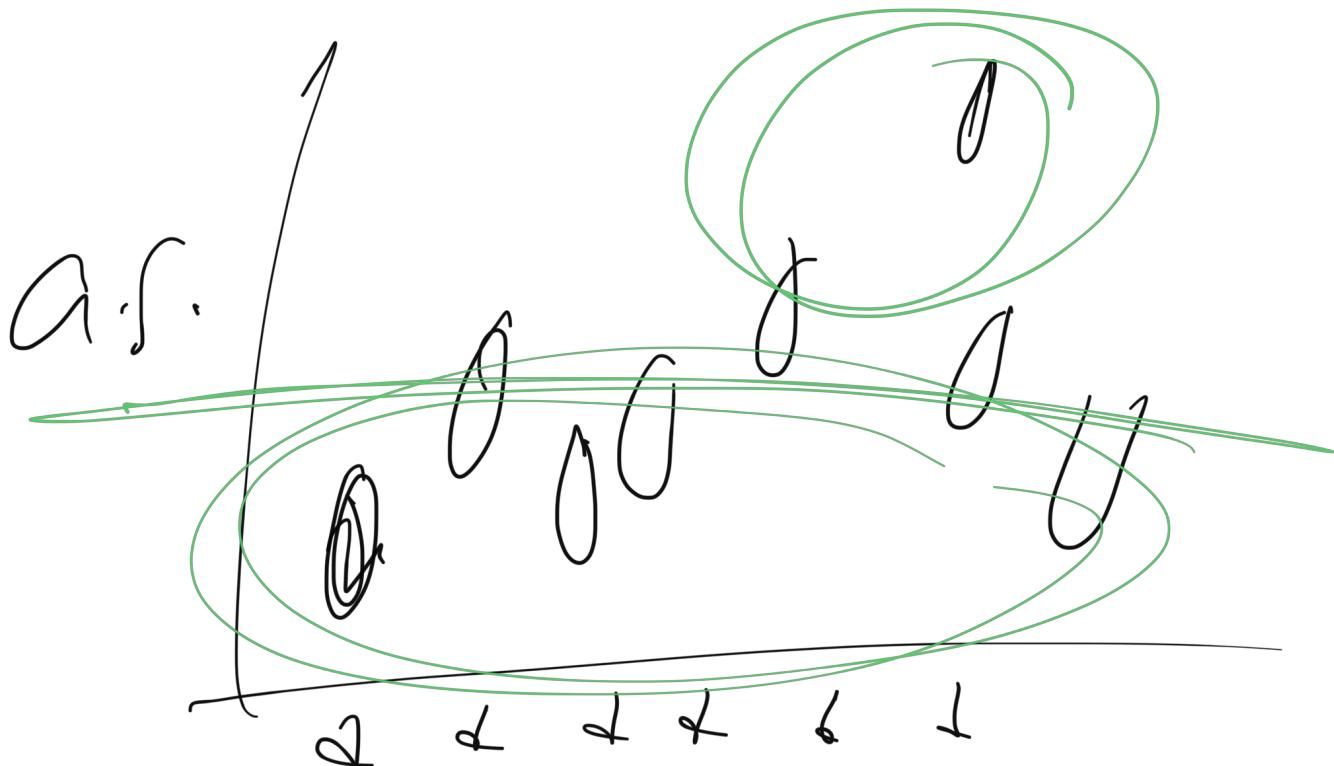




En las gráficas anteriores se puede ver que:

- Las edades de las mujeres casi hace una recta
- Los años de sentencia de las mujeres NO parecen normales sino más bien bimodales
- Las edades de los hombres SI parecen comportarse normales, ya que aparece la ralla roja.
- Los años de sentencia de los hombres NO parecen normales sino más bien bimodales

Otra prueba realizada es la de **Shapiro-Wilk**, la cual evalúa una muestra de datos y cuantifica la probabilidad de que los datos provienen de una distribución gaussiana o no. La función `shapiro()` devuelve el estadístico W calculado por la prueba como el valor p.



```
In [7]: # Shapiro-Wilk Test
import pandas as pd
from numpy.random import randn
from numpy.random import seed
from numpy import concatenate, isnan
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")

seed(7)
nd = 20 * randn(1000) + 35
datos = {'female_Edad': d.loc[d.gender == 'female'].age , \
          'female_Sentencia': d.loc[d.gender == 'female'].sentenceYears, \
          'male_edad': d.loc[d.gender == 'male'].age, \
          'male_Sentencia': d.loc[d.gender == 'male'].sentenceYears, \
          'unim': nd, \
          'bim': concatenate((nd, 10 * randn(1000) + 20))}

from scipy.stats import shapiro

for alpha in [0.05, 0.01]:
    for data in datos:
        da = datos[data]
        s, p = shapiro(da[~isnan(da)])
        print('{:s} {:.2f} {:.3f}'.format(data, s, p))
        if p > alpha:
            print('Si se ve gaussiana la muestra (no se puede rechazar H0), alpha: ', alpha)
        else:
            print('No se ve gaussiana la muestra (rechazar H0), alpha: ', alpha)
```

```
female_Edad 0.95 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.05
female_Sentencia 0.75 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.05
male_edad 0.94 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.05
male_Sentencia 0.82 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.05
unim 1.00 0.834
Si se ve gaussiana la muestra (no se puede rechazar H0), alpha:  0.05
bim 0.98 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.05
female_Edad 0.95 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.01
female_Sentencia 0.75 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.01
male_edad 0.94 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.01
male_Sentencia 0.82 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.01
unim 1.00 0.834
Si se ve gaussiana la muestra (no se puede rechazar H0), alpha:  0.01
bim 0.98 0.000
No se ve gaussiana la muestra (rechazar H0), alpha:  0.01
```

Con esta prueba de estadística se evidencia que nada mas pasan la prueba de normalidad son los datos auxiliares unimodales, por tanto se realizan además pruebas no paramétricas. La prueba de **Kruskal-Wallis** es una versión no paramétrica del análisis de varianza o ANOVA para abreviar, esta prueba se puede usar para determinar si más de dos muestras independientes tienen una distribución diferente. Cada muestra de datos debe ser independiente y pueden diferir en tamaño. En la siguiente prueba que se realizó se intenta comprobar que factores pueden influir en que se cometan determinados crímenes, los factores pueden ser la edad, el género, la educación, si tenía trabajo o no, y la cantidad de hijos.

```
In [59]: # Kruskal-Wallis H-test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import kruskal
from numpy import concatenate, isnan
import ssl
import pandas as pd

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")

from scipy.stats import kruskal

array = dict()

array['edad'] = [pedazo[1].crime.dropna() for pedazo in d.groupby(['age'])]
array['genero'] = [pedazo[1].crime.dropna() for pedazo in d.groupby(['gender'])]
array['educacion'] = [pedazo[1].crime.dropna() for pedazo in d.groupby(['education'])]
array['numero de hijos'] = [pedazo[1].crime.dropna() for pedazo in d.groupby(['numberOfChildren'])]
array['Tenia trabajo'] = [pedazo[1].crime.dropna() for pedazo in d.groupby(['workedBeforeFOAQ'])]

for datos in array:
    niv = array[datos]

    for alpha in [0.05, 0.01]:
        s, p = kruskal(*niv)
        print('{:s} {:s} {:.2f} {:.3f}'.format(datos, ' '.join([str(len(n)) for n in niv]), s, p))
        if p > alpha:
            print('No parece causar diferencia con nivel de significancia', alpha)
        else:
            print('hay por lo menos un factor que causa diferencia con nivel de significancia', alpha)

edad 9 0 2 5 6 18 17 11 14 26 16 20 18 17 17 20 13 15 16 15 12 17 16 18 7 17 10 9 11 3 12 6 2 9 3
2 6 5 3 2 2 2 1 3 1 2 2 2 1 1 4 1 1 0 0 nan nan
hay por lo menos un factor que causa diferencia con nivel de significancia 0.05
edad 9 0 2 5 6 18 17 11 14 26 16 20 18 17 17 20 13 15 16 15 12 17 16 18 7 17 10 9 11 3 12 6 2 9 3
2 6 5 3 2 2 2 1 3 1 2 2 2 1 1 4 1 1 0 0 nan nan
hay por lo menos un factor que causa diferencia con nivel de significancia 0.01
genero 167 302 2.13 0.144
No parece causar diferencia con nivel de significancia 0.05
genero 167 302 2.13 0.144
No parece causar diferencia con nivel de significancia 0.01
educacion 3 25 13 65 108 1 231 13 6.80 0.450
No parece causar diferencia con nivel de significancia 0.05
educacion 3 25 13 65 108 1 231 13 6.80 0.450
No parece causar diferencia con nivel de significancia 0.01
numero de hijos 20 76 114 100 53 24 5 3 2 1 1 2.79 0.986
No parece causar diferencia con nivel de significancia 0.05
numero de hijos 20 76 114 100 53 24 5 3 2 1 1 2.79 0.986
No parece causar diferencia con nivel de significancia 0.01
Tenia trabajo 141 23 0.35 0.553
No parece causar diferencia con nivel de significancia 0.05
Tenia trabajo 141 23 0.35 0.553
No parece causar diferencia con nivel de significancia 0.01
```

El único caso donde se ve diferencia es en la edad de los internos, con significancia de 0.01, entonces sí hay diferencia estadísticamente significativa en que la edad de las personas puede influir en el delito a cometer.

La prueba **Friedman Test** es la versión no paramétrica del análisis de la varianza del análisis de medidas repetidas o ANOVA de medidas repetidas. La prueba puede considerarse como una generalización de la Prueba H de Kruskal-Wallis a más de dos muestras. La hipótesis nula, indica que las muestras pareadas tienen la misma distribución. Un rechazo de la hipótesis nula indica que una o más de las muestras pareadas tienen una distribución diferente. Se implementa esta prueba utilizando la función `friedmanchisquare()`. Esta función toma como argumentos las muestras de datos para comparar y devuelve la estadística calculada y el valor p.

```
In [55]: # Friedman test
from numpy.random import seed
from numpy.random import randn
from scipy.stats import kruskal
from numpy import concatenate, isnan
import ssl
import pandas as pd

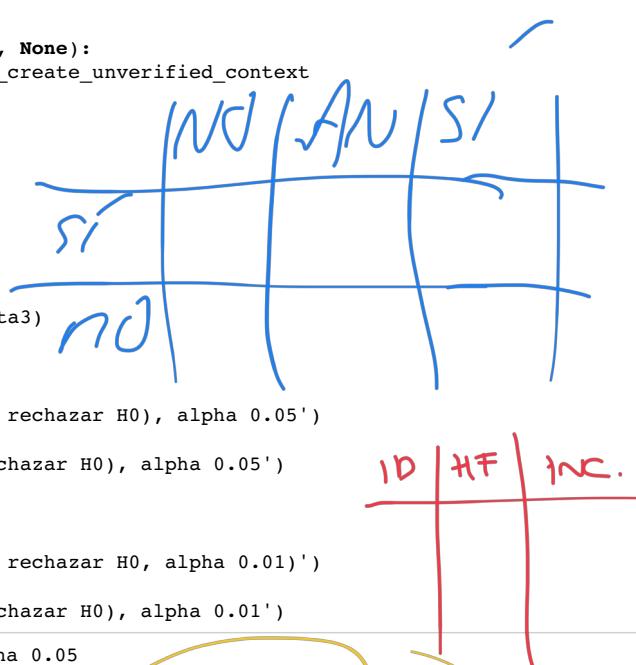
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica3ok.csv")

from scipy.stats import friedmanchisquare
# seed the random number generator
seed(1)
data1= d.sentenceYears
data2 = d.numberOfChildren
data3 = d.age

stat, p = friedmanchisquare(data1, data2, data3)

alpha = 0.05
if p > alpha:
    print('Las mismas distribuciones (no rechazar H0), alpha 0.05')
else:
    print('Diferentes distribuciones (rechazar H0), alpha 0.05')

alpha = 0.01
if p > alpha:
    print('Las mismas distribuciones (no rechazar H0, alpha 0.01)')
else:
    print('Diferentes distribuciones (rechazar H0), alpha 0.01')
```



1. Discución de la Práctica 6

La **Práctica 6** consiste en realizar modelos lineales con los datos. El objetivo de esta práctica consiste en modelar algunos aspectos de los datos recopilados de las encuestas con modelos lineales simples para concluir si los modelos que se obtienen son significativos o no, de este modo se puede comprobar si existe relación o dependencia entre algunos datos. Otro objetivo consiste en graficar los resultados obtenidos en estos modelos lineales. Los modelos lineales se realizan primero para los datos de los hombres y luego de las mujeres, con el objetivo de encontrar aspectos significativos que condicionen la necesidad de las personas de cometer delitos.

La tarea fue entregada en tiempo y calificada con calificación 6 ya que se penalizó un punto ya que los resultados de algunos de los modelos no fueron significativos.

Analisis-Estadistico-Multivariado (/github/EvelyGutierrez/Analisis-Estadistico-Multivariado/tree/master)
/
ReportePractica6_Evely.ipynb (/github/EvelyGutierrez/Analisis-Estadistico-Multivariado/tree/master/ReportePractica6_Evely.ip



REPORTE PRÁCTICA 6: Modelos lineales

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

El objetivo de esta práctica es modelar algunos aspectos de los datos recopilados de las encuestas con modelos lineales simples, concluir si los modelos que se obtienen son significativos o no y graficar los resultados.

Desarrollo

Se realizan modelos lineales separando los datos en dos grupos, mujeres y hombre, es decir, los modelos lineales se realizan primero para los datos de los hombres y luego de las mujeres, con el objetivo de encontrar aspectos significativos que condicionen la necesidad de las personas de cometer delitos.

Se filtra los años de sentencia de los internos hombres con edades mayores a 15 años.

```
In [69]: import pandas as pd
from numpy import isnan
from statsmodels.graphics.gofplots import qqplot
import matplotlib.pyplot as plt
from scipy.stats import shapiro
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica6.csv")
print(len(d))
e = d.loc[d.gender == 'male'] # filtrar por genero, hombres
print(len(e))
d1 = e.loc[e.age >= 16] # Mayores de 16 años de edad
print(len(d1))
d2 = e.loc[~isnan(e.sentenceYears)]
```

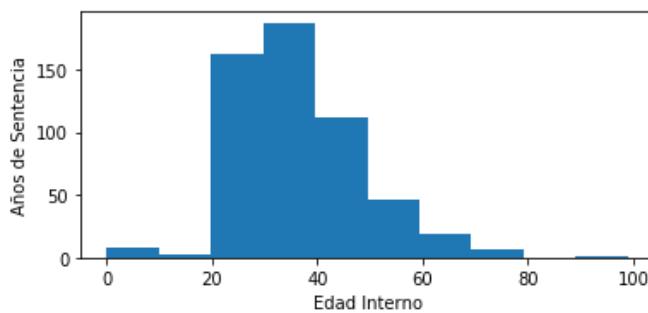
```
print(len(d2))
m = pd.concat([d1, d2])
print(len(m))
d = m.age.dropna()
```

```
plt.rcParams["figure.figsize"] = [6, 9]
f = plt.figure()

sf = f.add_subplot(312)
sf.hist(d)
sf.set_xlabel("Edad Interno")
sf.set_ylabel("Años de Sentencia")
```

```
484
312
300
243
543
```

Out[69]: Text(0, 0.5, 'Años de Sentencia')



En las gráficas anteriores se ve que en el caso de los hombres siguen una distribución aparentemente normal. Luego se comienzan modelar los años de sentencia como una función de la edad de los internos hombres.

```
In [62]: # Hombres internos, edad y años de sentencia
import matplotlib.pyplot as plt
from scipy import stats
from numpy import isnan
import pandas as pd
```

```

import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
e = o.loc[o.gender == 'male']
m = e.loc[~isnan(e.age) > 16]
x = m.age
y = m.sentenceYears

mascara = ~isnan(x) & ~isnan(y)
x = x[mascara]
y = y[mascara]

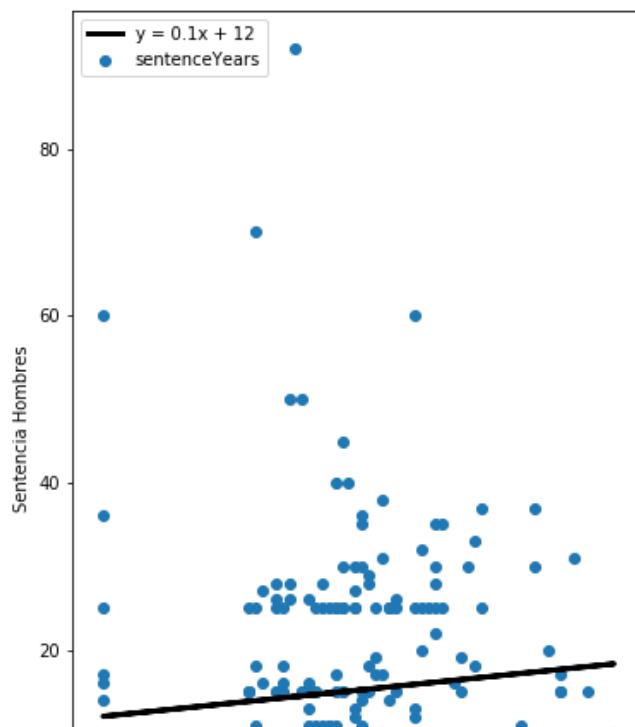
a, b, r, p, e = stats.linregress(x, y)
print("y = f(x) = {:.4f} x + {:.4f}".format(a, b))
print("error", e)
print("valor p", p)
print("pendiente {:s}significativo".format("no " if p >= 0.05 else ""))
print("R^2", r**2)

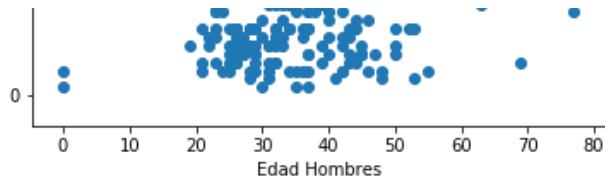
plt.plot(x, (a * x + b), label = 'y = {:.1f}x + {:.0f}'.format(a, b), color = 'black', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Edad Hombres")
plt.ylabel("Sentencia Hombres")

```

$y = f(x) = 0.0817 x + 12.0093$
 error 0.06292450975594598
 valor p 0.19544903608298309
 pendiente no significativo
 R^2 0.006944806502868976

Out[62]: Text(0, 0.5, 'Sentencia Hombres')





Se obtiene una pendiente no significativa, el error es pequeño y se nota que van aumentando los años de sentencia con la edad, o que la edad de los hombres condiciona de algún modo la cantidad de años que les impongan. A medida que van creciendo los años de edad, va creciendo levemente los años de sentencia.

Además, se trabaja con los ingresos mensuales y los años de sentencia de los internos hombres, dado que puede ser que tenga valor significativo el hecho de tener menos dinero a cometer delitos más o menos graves y por consecuencia mayor o menor cantidad de años. Los ingresos mensuales antes de entrar a la prisión se registran en la encuesta de la siguiente forma:

1. Menos de 3,000
2. De 3,001 a 6,000
3. De 6,001 a 9,000
4. Más de 9,000

Se intenta modelar los años de sentencia en función de la cantidad de dinero que tenían los internos hombres mensualmente.

```
In [71]: # Hombres internos, años de sentencia e ingresos mensuales
import matplotlib.pyplot as plt
from scipy import stats
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
e = o.loc[o.gender == 'male']
m = e.loc[~isnan(e.age)> 16]

y = m.sentenceYears # modelando
x = m.incomePrev # en función de  $y = f(x) = a * x + b$ 

mascara = ~isnan(x) & ~isnan(y)
x = x[mascara]
y = y[mascara]

a, b, r, p, e = stats.linregress(x, y)
print("y = f(x) = {:.4f} x + {:.4f}".format(a, b))
print("error", e)
print("valor p", p)
print("pendiente {:s}significativo".format("no " if p >= 0.05 else ""))
print("R^2", r**2)

plt.plot(x, (a * x + b), label = 'y = {:.1f}x + {:.0f}'.format(a, b), color = 'black', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper center')
plt.xlabel("Ingreso Mensual Hombres")
plt.xticks([1, 2, 3, 4], ["3000", "6000", "9000", "más de 9000"])
```

```

plt.ylabel("Años de sentencia Hombres")
plt.show()

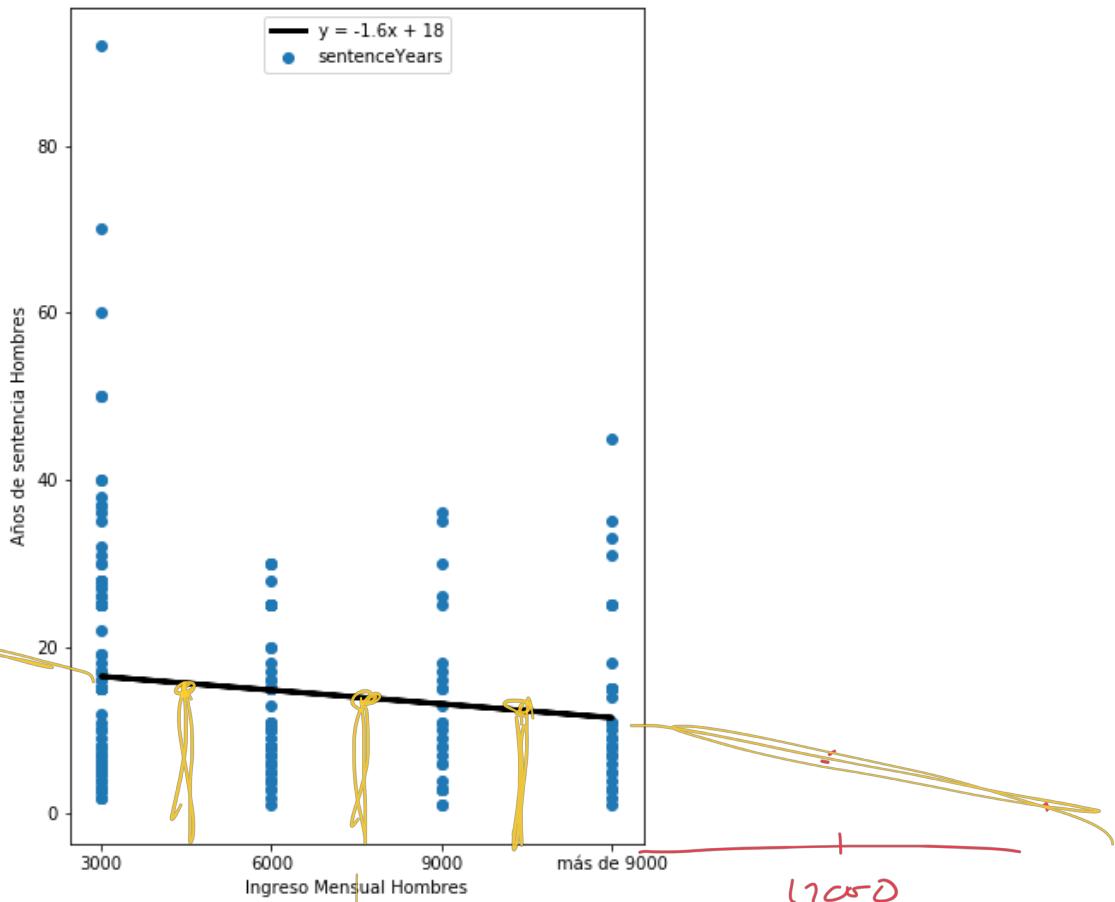
print("Si gana menos de 3000, aproximado sentencia {:.0f}".format(b))
print("Si gana más de 9000, aproximado sentencia {:.0f}".format(a + b))

```

```

y = f(x) = -1.6490 x + 18.0927
error 0.7573076672582115
valor p 0.030488607756852505
pendiente significativo
R^2 0.020637081133893283

```



Si gana menos de 3000, aproximado sentencia 18
 Si gana más de 9000, aproximado sentencia 16

Se obtiene una pendiente significativa, se nota que la pendiente baja a medida que más ingresos mensuales tenga el interno, parece que mientras más ingresos mensuales tengan los hombres, el delito a cometer es más leve, o pudiera decirse que los años de sentencia de los hombres se pueden estimar como -1,6 veces del ingreso mensual más 18, aproximadamente.

Se realiza estos mismos trabajos con las mujeres internas, primero se investiga la relación entre los años de sentencia que con la edad de las internas usando la misma metodología.

```

In [59]: # Mujeres internas, edad y años de sentencia
import matplotlib.pyplot as plt
from scipy import stats

```

```

from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
e = o.loc[o.gender == 'female']
m = e.loc[~isnan(e.age) > 16]
x = m.age
y = m.sentenceYears

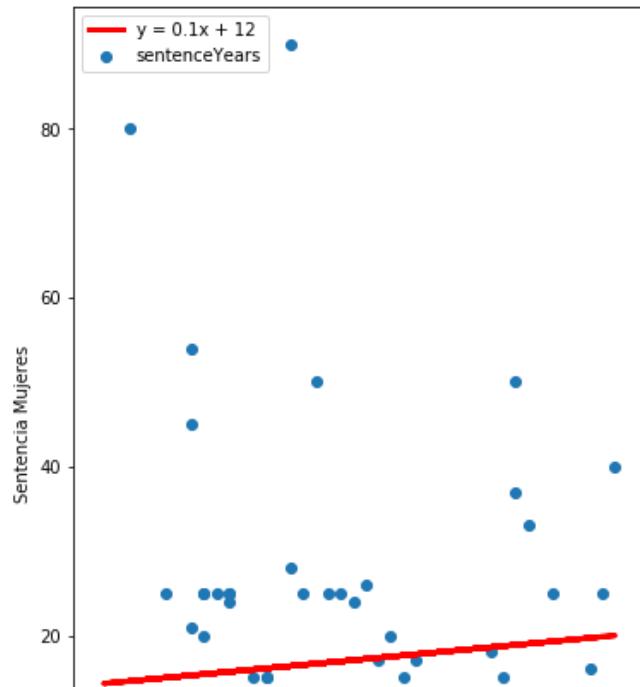
mascara = ~isnan(x) & ~isnan(y)
x = x[mascara]
y = y[mascara]

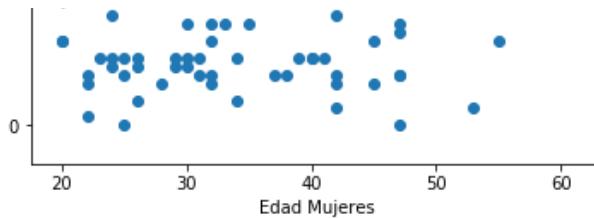
a, b, r, p, e = stats.linregress(x, y)
print("y = f(x) = {:.4f} x + {:.4f}".format(a, b))
print("error", e)
print("valor p", p)
print("pendiente {:s}significativo".format("no " if p >= 0.05 else ""))
print("R^2", r**2)

plt.plot(x, (a * x + b), label = 'y = {:.1f}x + {:.0f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper left')
plt.xlabel("Edad Mujeres")
plt.ylabel("Sentencia Mujeres")
y = f(x) = 0.1386 x + 11.5369
error 0.16527383815723537
valor p 0.40422363964375885
pendiente no significativo
R^2 0.00829724724138278

```

Out[59]: Text(0, 0.5, 'Sentencia Mujeres')





Se obtiene pendiente no significativa en este caso, se prueba entonces con los años de sentencia y su relación con los ingresos mensuales de las mujeres.

```
In [60]: # Mujeres internas, años de sentencia e ingresos mensuales
import matplotlib.pyplot as plt
from scipy import stats
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
e = o.loc[o.gender == 'female']
m = e.loc[e.age >= 16]

y = m.sentenceYears # modelando
x = m.incomePrev # en función de  $y = f(x) = a * x + b$ 

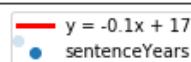
mascara = ~isnan(x) & ~isnan(y)
x = x[mascara]
y = y[mascara]

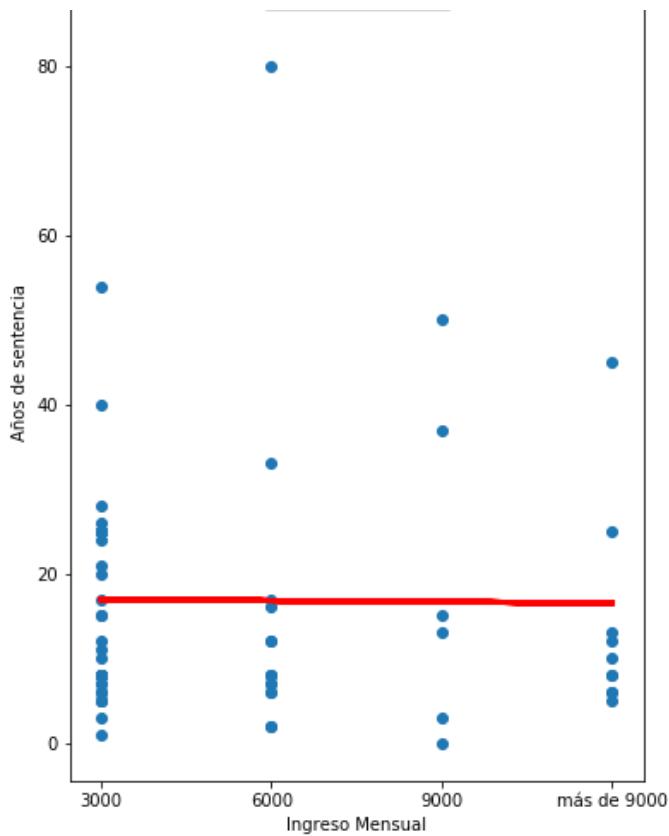
a, b, r, p, e = stats.linregress(x, y)
print("y = f(x) = {:.4f} x + {:.4f}".format(a, b))
print("error", e)
print("valor p", p)
print("pendiente {:s}significativo".format("no " if p >= 0.05 else ""))
print("R^2", r**2)

plt.plot(x, (a * x + b), label = 'y = {:.1f}x + {:.0f}'.format(a, b), color = 'red', linewidth = 3)
plt.scatter(x, y)
plt.legend(loc='upper center')
plt.xlabel("Ingreso Mensual")
plt.xticks([1, 2, 3, 4], ["3000", "6000", "9000", "más de 9000"])
plt.ylabel("Años de sentencia")
plt.show()

print("Si gana menos de 3000, aproximado sentencia {:.0f}".format(b))
print("Si gana más de 9000, aproximado sentencia {:.0f}".format(a + b))
```

y = f(x) = -0.1476 x + 17.0992
 error 1.8303303884164583
 valor p 0.9359475964444619
 pendiente no significativo
 R² 9.427632609560544e-05





Si gana menos de 3000, aproximado sentencia 17

Si gana más de 9000, aproximado sentencia 17

De igual modo la pendiente no es significativa para esta relación en el caso de las mujeres internas, esto puede ser debido a que los datos de las mujeres al parecer no están cerca de ser normalmente distribuidos, por tanto, vamos a revisar como se encuentran estos datos en las mujeres.

```
In [63]: import pandas as pd
from numpy import isnan
from statsmodels.graphics.gofplots import qqplot
import matplotlib.pyplot as plt
from scipy.stats import shapiro
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica6.csv")
print(len(d))
e = d.loc[d.gender == 'female'] # filtrar por genero, mujeres
print(len(e))
d1 = e.loc[e.age >= 16] # Mayores de 16 años de edad
print(len(d1))
d2 = e.loc[~isnan(e.sentenceYears)] 

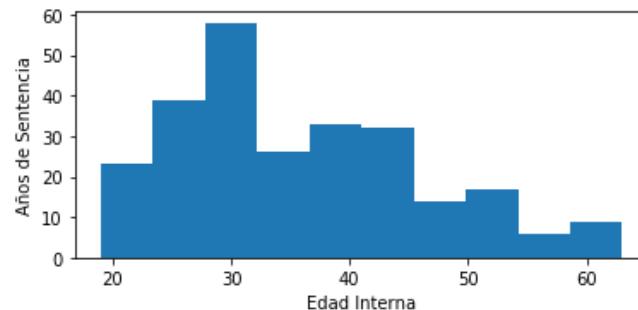
print(len(d2))
m = pd.concat([d1, d2])
print(len(m))
d = m.age.dropna()

plt.rcParams["figure.figsize"] = [6, 9]
f = plt.figure()

sf = f.add_subplot(312)
sf.hist(d)
sf.set_xlabel("Edad Interna")
sf.set_ylabel("Años de Sentencia")
```

484
172
171
86
257

Out[63]: Text(0, 0.5, 'Años de Sentencia')



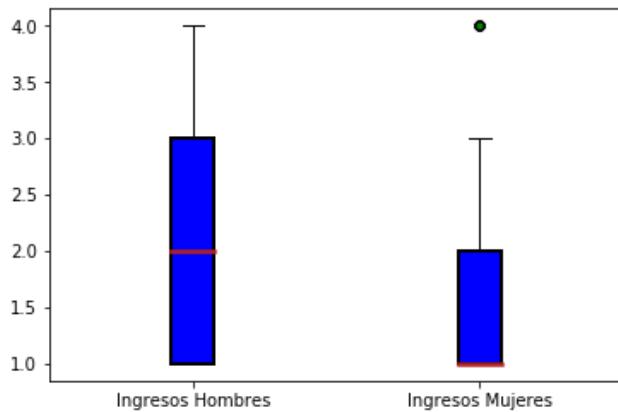
Como se ve en las gráficas anteriores la distribución de los datos de las mujeres no están nada normales, pudiendo ser esto la causa de los resultados no significativos obtenidos anteriormente.

```
In [34]: from scipy.stats import mannwhitneyu
import matplotlib.pyplot as plt
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica6.csv")
hombre = d.loc[d.gender == 'male']
mujer = d.loc[d.gender == 'female']
h = hombre.incomePrev.dropna()
m = mujer.incomePrev.dropna()
print(mannwhitneyu(h, m))

bp = dict(linestyle='-', linewidth = 2, color='black', facecolor='blue')
fp = dict(marker = 'o', markerfacecolor = 'green', markersize = 5, linestyle = 'none')
mp = dict(linestyle = '-', linewidth = 2.5, color ='firebrick')
plt.boxplot([h, m], labels=["Ingresos Hombres", "Ingresos Mujeres"], \
            boxprops = bp, flierprops = fp, medianprops = mp, patch_artist = True)
plt.show()
```

MannwhitneyuResult(statistic=18191.5, pvalue=0.008141761326959098)



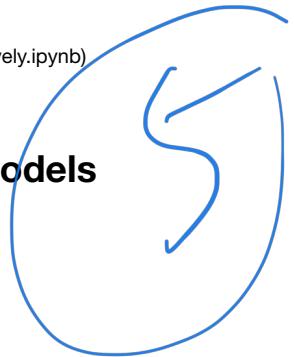
El diagrama anterior muestra una diferencia un poco más significativa en cuanto a los ingresos mensuales entre los hombres y mujeres, siendo las mujeres las de menos ingresos, pudiendo ser esto una de las causas probables de llevarlas a cometer delitos.

En resumen, ninguno de los modelos anteriores fue muy bueno, los datos no se encuentran normalmente distribuidos y esto no ayuda los modelos. A pesar de esto se pudo apreciar que en el caso de los hombres puede existir una relación de dependencia de los años de sentencia con respecto a la edad de los internos, y un poco más con la cantidad de dinero que tenían como ingreso mensual, cosa que en el caso de las mujeres no se comportó de esta manera, por lo cual valdría la pena seguir revisando que factores condicionan el hecho de que las mujeres cometan ciertos delitos.

1. Discución de la Práctica 7

La **Práctica 7** consiste en trabajos de regresión múltiple con statsmodels. El objetivo de esta práctica consiste en trabajar con modelos de regresión múltiple y usar los resultados obtenidos como un clasificador de alguna variable de interés. En esta práctica los modelos lineales tienen más de un factor para poder modelar conjuntamente la variable de interés. Se visualizan los resultados obtenidos en gráficas de dispersión para comprobar las correlaciones que puedan existir entre los datos de interés.

La tarea fue entregada en tiempo y calificada con calificación 5 y se penalizaron 2 puntos ya que los resultados obtenidos en algunos modelos no fueron significativos.



REPORTE PRÁCTICA 7: Regresión múltiple con statsmodels

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

El objetivo de este reporte es trabajar con modelos de regresión múltiple y usar los resultados obtenidos como un clasificador de alguna variable de interés.

Esta práctica se continúa trabajando sobre los mismos datos de la anterior, pero en esta los modelos lineales tienen más de un factor para poder modelar conjuntamente la variable de interés. Se agregaron dos columnas nuevas para trabajar con las columnas de crimen y educación de los internos, ya que estas son de tipo **string** y se categorizaron para poderlas utilizar en los modelos de regresión múltiple. A continuación, se muestran estas dos columnas y sus valores categóricos.

```
In [121]: import pandas as pd
import numpy as np
import statsmodels.api as sm

df = pd.read_csv('practica6.csv', index_col=0)
X = df.copy()
y = X.pop('education')

df.head()
print(y.groupby(X.educacion).unique())

X = df.copy()
y = X.pop('crimen')

df.head()
print(y.groupby(X.crimen).unique())
```

```
educacion
Comercial      [6.0]
Licenciatura   [5.0]
Preparatoria   [3.0]
Primaria       [1.0]
Secretaria     [7.0]
Secundaria     [2.0]
Tecnico        [4.0]
Name: education, dtype: object
crimen
Accidente      [9.0]
Armas          [8.0]
Atentado al pudor [11.0]
Chantaje       [12.0]
Contra la salud [6.0]
Dano a instituciones publicas [10.0]
Dano en propiedad ajena [10.0]
Delicuencia organizada [13.0]
Despojo de inmueble [14.0]
Drogas          [9.0]
Drogas y Armas [5.0]
Empapelado     [15.0]
Federal        [16.0]
Fraude          [17.0]
Homicidio      [2.0]
Lesiones        [18.0]
Narcomenudeo   [19.0]
Pension alimenticia [20.0]
Privacion de la libertad [21.0]
Robo            [1.0]
Secuestro       [7.0]
Trata de personas [22.0]
Violacion      [3.0]
Violencia Familiar [4.0]
Name: crime, dtype: object
```

El siguiente código analiza los valores de las columnas de crimen, educación, años de sentencia e ingresos mensuales antes de entrar a la prisión, en función de la edad de los internos, primero para los hombres y luego para las mujeres.

```
In [129]: import statsmodels.api as sm
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
male = x.loc[x.age >= 16] # Hombres
x = o.loc[o.gender == 'female']
female = x.loc[x.age >= 16] # Mujeres
```

```

for datos in [male, female]:
    d = pd.DataFrame(datos, columns = ["crime", "age", "sentenceYears", "education", "incomePrev"])
    d = d.dropna() # ignoraremos renglones que contienen por lo menos un NaN para estos campos
    n = len(d)
    if n >= 8: # no se puede con menos de ocho
        y = d["age"]
        x = d[["crime", "sentenceYears", "education", "incomePrev"]]
        x = sm.add_constant(x) # para contar con la b en nuestra f()
        m = sm.OLS(y, x).fit()
        print(datos.gender.unique()[0])
        print(m.summary())

```

male

OLS Regression Results

Dep. Variable:	age	R-squared:	0.062			
Model:	OLS	Adj. R-squared:	0.043			
Method:	Least Squares	F-statistic:	3.299			
Date:	Tue, 12 Mar 2019	Prob (F-statistic):	0.0121			
Time:	16:41:46	Log-Likelihood:	-763.07			
No. Observations:	204	AIC:	1536.			
Df Residuals:	199	BIC:	1553.			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	29.8322	2.225	13.409	0.000	25.445	34.220
crime	0.1394	0.164	0.851	0.396	-0.184	0.463
sentenceYears	0.1510	0.056	2.678	0.008	0.040	0.262
education	0.3894	0.788	0.494	0.622	-1.164	1.943
incomePrev	1.6637	0.713	2.335	0.021	0.258	3.069
Omnibus:	19.331	Durbin-Watson:			1.481	
Prob(Omnibus):	0.000	Jarque-Bera (JB):			21.838	
Skew:	0.768	Prob(JB):			1.81e-05	
Kurtosis:	3.459	Cond. No.			63.4	

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
female

OLS Regression Results

Dep. Variable:	age	R-squared:	0.130			
Model:	OLS	Adj. R-squared:	0.075			
Method:	Least Squares	F-statistic:	2.351			
Date:	Tue, 12 Mar 2019	Prob (F-statistic):	0.0636			
Time:	16:41:46	Log-Likelihood:	-249.38			
No. Observations:	68	AIC:	508.8			
Df Residuals:	63	BIC:	519.9			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
const	28.3616	3.766	7.531	0.000	20.836	35.887
crime	-0.1262	0.270	-0.467	0.642	-0.666	0.414
sentenceYears	0.0227	0.072	0.317	0.752	-0.120	0.166
education	1.2773	0.921	1.387	0.170	-0.563	3.118
incomePrev	2.4693	1.146	2.154	0.035	0.179	4.760
Omnibus:	4.276	Durbin-Watson:			1.672	
Prob(Omnibus):	0.118	Jarque-Bera (JB):			4.031	
Skew:	0.595	Prob(JB):			0.133	
Kurtosis:	2.908	Cond. No.			77.2	

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Para el caso de la variable crimen, por ejemplo, se puede ver que este modelo tiene un valor de R cuadrado o porcentaje de varianza: 0.062 y 0.130 (hombres y mujeres), lo que significa que estos modelos explican unos porcentajes muy bajos de la varianza en la variable dependiente. Se puede ver que ninguna de las variables crimen, educación, años de sentencia e ingresos mensuales son estadísticamente significativas para predecir (o estimar) la edad de los internos.

El siguiente código verifica la correlación entre los diferentes tipos de crímenes y las variables edad, años de sentencia, ingresos mensuales y nivel de educación, donde únicamente se vio más correlación en el delito del tipo Violación en el caso de las mujeres.

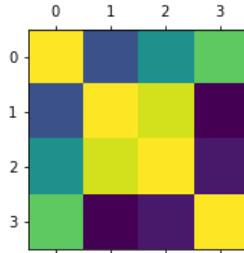
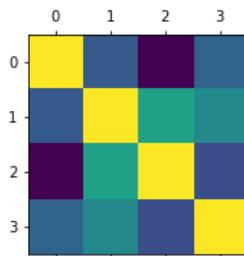
```
In [147]: import statsmodels.api as sm
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
male = x.loc[x.crimen == 'Violacion'] # Hombres
x = o.loc[o.gender == 'female']
female = x.loc[x.crimen == 'Violacion'] # Mujeres

f = plt.figure()
d = male
d = pd.DataFrame(d, columns = ["education", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(d.corr())
sf = f.add_subplot(121)
tmp = sf.matshow(d.corr())

f = plt.figure()
d = female
d = pd.DataFrame(d, columns = ["education", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(d.corr())
sf = f.add_subplot(121)
tmp = sf.matshow(d.corr())
```

	education	age	sentenceYears	incomePrev
education	1.000000	0.165115	-0.165114	0.210475
age	0.165115	1.000000	0.500257	0.381184
sentenceYears	-0.165114	0.500257	1.000000	0.116058
incomePrev	0.210475	0.381184	0.116058	1.000000
	education	age	sentenceYears	incomePrev
education	1.0	-0.500000	0.000000	0.500000
age		-0.5	1.000000	-1.000000
sentenceYears		0.0	0.866025	1.000000
incomePrev		0.5	-1.000000	-0.866025

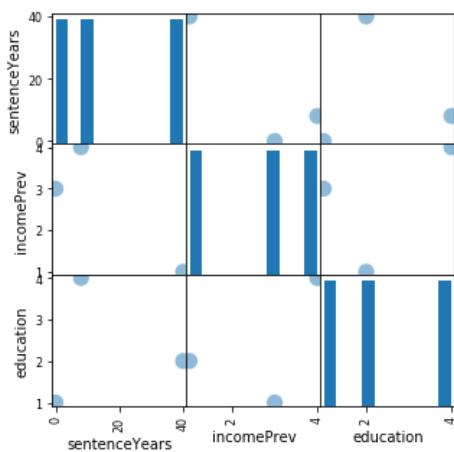


Tanto para los hombres y las mujeres, correlacionaron bastante parecidas las variables crimen, educación, edad, años de sentencia e ingresos mensuales antes de entrar a la prisión. En el caso de las mujeres que cometan el delito de Violación, si se ve un poco más de relación, entonces se realiza una gráfica de dispersión para revisar lo antes visto.

```
In [187]: import matplotlib.pyplot as plt
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'female']
d = x.loc[x.crimen == 'Violacion']
d = pd.DataFrame(d, columns = ["sentenceYears", "incomePrev", "education"])
d = d.dropna()
print(len(d))
tmp = pd.plotting.scatter_matrix(d, figsize = (5,5), s = 500)
```

3



No se puede concluir muchas cosas ya que son 3 mujeres nada más. Se prueban otras relaciones entre las variables para ver cómo se comportan.

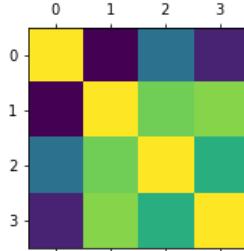
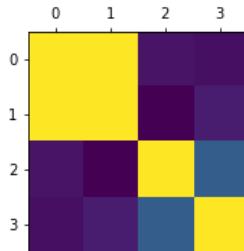
```
In [193]: import statsmodels.api as sm
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
male = x.loc[x.educacion == 'Tecnico'] # Hombres
x = o.loc[o.gender == 'female']
female = x.loc[x.educacion == 'Tecnico'] # Mujeres

f = plt.figure()
d = male
d = pd.DataFrame(d, columns = ["crime", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(d.corr())
sf = f.add_subplot(121)
tmp = sf.matshow(d.corr())

f = plt.figure()
d = female
d = pd.DataFrame(d, columns = ["crime", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(d.corr())
sf = f.add_subplot(121)
tmp = sf.matshow(d.corr())
```

	crime	age	sentenceYears	incomePrev
crime	1.000000	0.993884	-0.554700	-0.577350
age	0.993884	1.000000	-0.643192	-0.510061
sentenceYears	-0.554700	-0.643192	1.000000	-0.160128
incomePrev	-0.577350	-0.510061	-0.160128	1.000000
	crime	age	sentenceYears	incomePrev
crime	1.000000	-0.902421	-0.192613	-0.717593
age	-0.902421	1.000000	0.587454	0.654177
sentenceYears	-0.192613	0.587454	1.000000	0.294331
incomePrev	-0.717593	0.654177	0.294331	1.000000



Se vuelve a probar las correlaciones entre las variables utilizadas anteriormente, pero esta vez con el nivel de escolaridad, resultando los datos con mayor correlación para el caso de los hombres y mujeres con nivel de educación Técnico. Se realiza un diagrama de dispersión para verificar.

```
In [194]: import matplotlib.pyplot as plt
```

```

from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
d = x.loc[x.educacion == 'Tecnico']
d = pd.DataFrame(d, columns = ["sentenceYears", "incomePrev", "crime"])
d = d.dropna()
print(len(d))
tmp = pd.plotting.scatter_matrix(d, figsize = (5,5), s = 500)

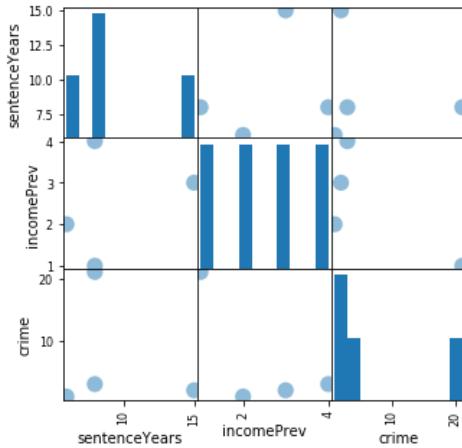
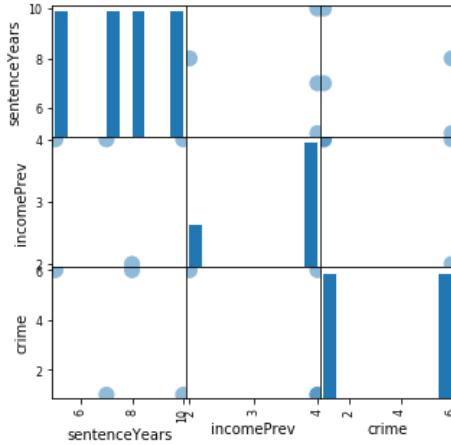
import matplotlib.pyplot as plt
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'female']
d = x.loc[x.educacion == 'Tecnico']
d = pd.DataFrame(d, columns = ["sentenceYears", "incomePrev", "crime"])
d = d.dropna()
print(len(d))
tmp = pd.plotting.scatter_matrix(d, figsize = (5,5), s = 500)

```

4

4



Para el caso de las mujeres y hombres no se puede concluir mucho ya que solo son 4. Se intenta realizar el mismo trabajo, pero ahora para el nivel de educación Secundaria ya que en prácticas anteriores se vio que la mayoría de los internos tienen este nivel de educación.

```
In [198]: import statsmodels.api as sm
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
male = x.loc[x.educacion == 'Secundaria'] # Hombres
x = o.loc[o.gender == 'female']
female = x.loc[x.educacion == 'Secundaria'] # Mujeres

f = plt.figure()
d = male
d = pd.DataFrame(d, columns = ["crime", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(d.corr())
sf = f.add_subplot(121)
tmp = sf.matshow(d.corr())

f = plt.figure()
d = female
d = pd.DataFrame(d, columns = ["crime", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(d.corr())
sf = f.add_subplot(121)
tmp = sf.matshow(d.corr())

import matplotlib.pyplot as plt
from numpy import isnan
import pandas as pd
import ssl

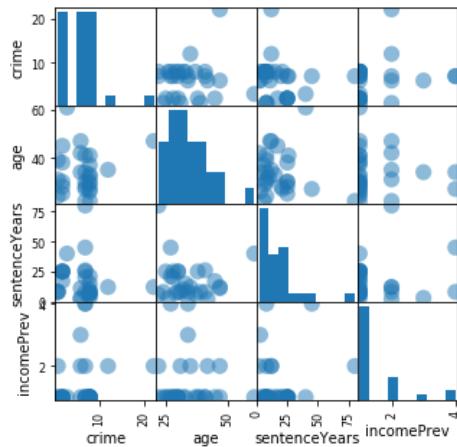
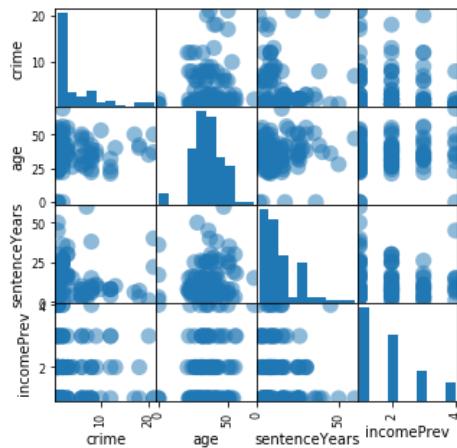
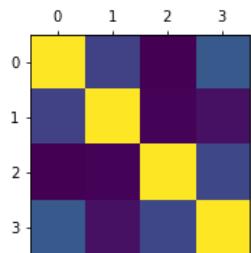
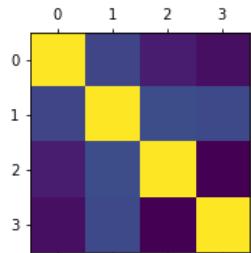
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
d = x.loc[x.educacion == 'Secundaria']
d = pd.DataFrame(d, columns = ["crime", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(len(d))
tmp = pd.plotting.scatter_matrix(d, figsize = (5,5), s = 500)

import matplotlib.pyplot as plt
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'female']
d = x.loc[x.educacion == 'Secundaria']
d = pd.DataFrame(d, columns = ["crime", "age", "sentenceYears", "incomePrev"])
d = d.dropna()
print(len(d))
tmp = pd.plotting.scatter_matrix(d, figsize = (5,5), s = 500)
```

	crime	age	sentenceYears	incomePrev
crime	1.000000	0.109417	-0.035970	-0.079961
age	0.109417	1.000000	0.143207	0.123853
sentenceYears	-0.035970	0.143207	1.000000	-0.124512
incomePrev	-0.079961	0.123853	-0.124512	1.000000

	crime	age	sentenceYears	incomePrev
crime	1.000000	0.104847	-0.111782	0.199245
age	0.104847	1.000000	-0.103000	-0.062920
sentenceYears	-0.111782	-0.103000	1.000000	0.129436
incomePrev	0.199245	-0.062920	0.129436	1.000000
116				
29				



Se ve en las gráficas anteriores que la correlación entre las variables crimen, años de sentencia, edad e ingresos mensuales con respecto al nivel de educación Secundaria no es muy alta, pero la cantidad de internos que tienen alguna relación en las variables analizadas es de 116 hombres y 29 mujeres.

1. Discución de la Práctica 8

La **Práctica 8** consiste en realizar análisis de varianzas y de componentes principales a los datos. El objetivo consiste en realizar un análisis de varianza utilizando ANOVA. Además se realiza un PCA en dos dimensiones con el objetivo de separar los hombres de las mujeres internas como dos clases de una forma clara. También se visualizan y discuten los resultados del ANOVA y del PCA.

La tarea fue entregada en tiempo y calificada con calificación 6 ya que se debió haber normalizado los datos numéricos utilizados para poder obtener mejores resultados.

REPORTE PRÁCTICA 8: Análisis de varianza y de componentes principales

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

En esta práctica se busca el objetivo de realizar un análisis de varianza utilizando [ANOVA](https://pythonfordatascience.org/anova-python/) (<https://pythonfordatascience.org/anova-python/>) e intentar realizar un **PCA** en dos dimensiones para ver si se logran separar los hombres de las mujeres internas como dos clases de una forma clara. Además, se visualizan y discuten los resultados del **ANOVA** y del **PCA**.

ANOVA es un proceso estadístico para analizar la cantidad de varianza que se contribuye a una muestra por diferentes factores. La hipótesis nula sería que no haya diferencias, es decir, si el valor p de una variable es menor a la significancia establecida (trabajando con un Alpha 0.05), entonces se rechaza la hipótesis nula, concluyendo que esa variable sí tiene un efecto estadísticamente significativo.

```
In [4]: import statsmodels.api as sm
from statsmodels.formula.api import ols
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practicac6.csv")
x = o.loc[o.gender == 'male'] # Para los internos Hombres
datos = pd.concat([x.loc[x.age >= 16], x.loc[~isnan(x.age)]])
d = pd.DataFrame(datos, columns = ["sentenceYears", "age", "incomePrev"])
d = d.dropna()
m = ols('sentenceYears ~ age + incomePrev', data = d).fit()
a = sm.stats.anova_lm(m, typ = 2)
print(a)
n = len(a)
alpha = 0.05
for i in range(n):
    print("{:s} {:s} es significativo".format(a.index[i], "" if a['PR(>F)'][i] < alpha else "no "))
    
```

"No es significativo"

	sum_sq	df	F	PR(>F)
age	1548.702158	1.0	10.091688	0.001593
incomePrev	1680.636422	1.0	10.951401	0.001011
Residual	68291.096829	445.0	NaN	NaN

age es significativo
incomePrev es significativo
Residual no es significativo

A pesar de que las variables son significativas, el valor de los residuos es muy alto, por tanto, se agregan otras columnas de datos que puedan marcar diferencias:

- nivel de educación
- tipo de crimen cometido
- estado civil del interno
- si tienen hijos o no
- si es primera vez o no que están cumpliendo condena.

```
In [5]: import statsmodels.api as sm
from statsmodels.formula.api import ols
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male'] # Para los internos Hombres
datos = pd.concat([x.loc[x.age >= 16], x.loc[~isnan(x.age)]])
d = pd.DataFrame(datos, columns = ["sentenceYears", "age", "incomePrev", "education", "crime", "prevRelStatus", "hasChildren", "firstOffense"])
d = d.dropna()
m = ols('sentenceYears ~ age + incomePrev + education + crime + prevRelStatus + hasChildren + firstOffense', data = d).fit()
a = sm.stats.anova_lm(m, typ = 2)
print(a)
n = len(a)
alpha = 0.05
for i in range(n):
    print("{}:{} {}es significativo".format(a.index[i], " if a['PR(>F)'][i] < alpha else "NO "))

      sum_sq      df          F      PR(>F)
age        1366.109398    1.0  1.157860e+01  0.000870
incomePrev     0.000115    1.0  9.724117e-07  0.999215
education      613.501213    1.0  5.199791e+00  0.024100
crime         1521.854335    1.0  1.289863e+01  0.000454
prevRelStatus    82.440547    1.0  6.987331e-01  0.404634
hasChildren      54.282905    1.0  4.600802e-01  0.498706
firstOffense      479.518822    1.0  4.064210e+00  0.045714
Residual       16518.005357   140.0           NaN      NaN
age es significativo
incomePrev NO es significativo
education es significativo
crime es significativo
prevRelStatus NO es significativo
hasChildren NO es significativo
firstOffense es significativo
Residual NO es significativo
```

En este caso bajó notablemente el valor residual, y tres de las variables resultaron ser significativas, la edad, la educación y el crimen cometido. Es interesante ver las interacciones entre estos factores para ver cómo se comportan.

```
In [3]: import statsmodels.api as sm
from statsmodels.formula.api import ols
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
datos = pd.concat([x.loc[x.age >= 16], x.loc[~isnan(x.age)]])
d = pd.DataFrame(datos, columns = ["sentenceYears", "age", "incomePrev", "education", "crime", "prevReStatus", "hasChildren", "firstOffense"])

d = d.dropna()
m = ols('sentenceYears ~ age * education + age * crime + education * crime ', \
        data = d).fit()
a = sm.stats.anova_lm(m, typ = 2)
print(a)
n = len(a)
alpha = 0.05
for i in range(n):
    print("{}:{} es significativo".format(a.index[i], "" if a['PR(>F)'][i] < alpha else "NO "))


```

	sum_sq	df	F	PR(>F)
age	1609.157272	1.0	13.976287	0.000268
education	812.769656	1.0	7.059287	0.008795
age:education	505.467586	1.0	4.390224	0.037934
crime	1085.692416	1.0	9.429749	0.002562
age:crime	87.376995	1.0	0.758910	0.385151
education:crime	246.112154	1.0	2.137600	0.145952
Residual	16234.008946	141.0	NaN	NaN
age es significativo				
education es significativo				
age:education es significativo				
crime es significativo				
age:crime NO es significativo				
education:crime NO es significativo				
Residual NO es significativo				

Según lo anterior parece haber interacción solo en la edad y la educación de los internos hombres en este caso.

Con el objetivo de tratar de reducir múltiples variables de entrada en una menor cantidad para modelar la variable de interés se realiza un análisis de componentes principales (PCA). Primeramente, se realizan algunas modificaciones en los datos que se tienen y se guardan en un fichero nuevo .csv, y se trabaja solo con las columnas que convienen codificar con números. Se estandarizan rangos para los valores que se van a tomar en cuenta y se visualizan los datos finales que quedan en el fichero resultante. Los datos estandarizados son los siguientes:

- Crimen
- Años de sentencia
- Educación
- Edad
- Si tenían trabajo o no antes de cometer un delito
- Si es primera vez o no que están privados de la libertad
- Los ingresos mensuales antes de estar privados de la libertad

```
In [10]: import ssl
import pandas as pd
from numpy import isnan, nan
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
x = o.loc[o.gender == 'male']
d = pd.concat([x.loc[x.age >= 16], x.loc[~isnan(x.age)]])
print('Qué tenemos')
```

```

print(d.columns)
d = d.drop(['Unnamed: 0', 'Unnamed: 0.1',
            'gender', 'hasChildren', 'numberOfChildren',
            'howManyBornInPrisonFOAQ', 'childrenLiveInPrisonFOAQ',
            'childrenSegPopFOAQ', 'childrenSocialSecurityFOAQ', 'prevRelStatus',
            'crimen','educacion'], 1)

print('Qué categorizamos tal cual')
for c in ['crime', 'sentenceYears', 'education',
           'age', 'workedBeforeFOAQ', 'firstOffense', 'incomePrev']:
    cat = pd.Categorical(d[c])
    d[c] = cat.codes

for c in d.columns:
    print(d[c].unique(), d[c].dtype)
xVars = ['crime', 'sentenceYears', 'education',
         'age', 'workedBeforeFOAQ', 'firstOffense', 'incomePrev']

d = d.dropna()
x = d.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
y1 = d.loc[:,['age']].values
y2 = d.loc[:,['sentenceYears']].values
k = 2 # Para dos dimensiones
pca = PCA(n_components = k)
cd = pd.DataFrame(data = pca.fit_transform(x), columns = ['comp_{:d}'.format(i) for i in range(k)])
cd['age'] = y1
cd['sentenceYears'] = y2
ordenado = pd.DataFrame.sort_values(cd, ['age'], ascending = False)
display(cd.head(10))
cd.to_csv('practica7.csv')

```

Qué tenemos

```

Index(['Unnamed: 0', 'Unnamed: 0.1', 'crime', 'sentenceYears', 'education',
       'age', 'gender', 'hasChildren', 'numberOfChildren',
       'howManyBornInPrisonFOAQ', 'childrenLiveInPrisonFOAQ',
       'childrenSegPopFOAQ', 'childrenSocialSecurityFOAQ', 'prevRelStatus',
       'workedBeforeFOAQ', 'firstOffense', 'incomePrev', 'crimen',
       'educacion'],
      dtype='object')

```

Qué categorizamos tal cual

```

[ 0   1   2   3   18  4   5   7   6   16  8   -1  11  14  19  17  12  13  9   15  10] int8
[ 5   34  31  7   24  35  16  18  21  1   14  3   0   -1  10  4   6   15  2   11  23  9   17  27
  8   36  30  12  19  26  38  13  33  28  39  22  32  37  20  29  25] int8
[ 1   0   2   -1  3   4   5] int8
[11  18  22  15  8   23  19  24  37  30  14  5   17  16  6   9   31  13  35  26  7   25  10  28
 21  1   36  27  33  4   2   12  20  51  29  38  41  44  34  47  43  42  45  50  40  49  48  39
 46  32  3   0] int8
[-1] int8
[ 0   1   -1] int8
[ 1   0   2   3   -1] int8

```

```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:
  Data with input dtype int8 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:
  Data with input dtype int8 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)

```

	comp_0	comp_1	age	sentenceYears
0	0.157177	-0.590874	11	5
1	-2.105771	1.053757	18	34
2	-0.898662	1.563234	22	31
3	0.665591	-0.073561	15	7
4	0.675029	-0.367518	8	7
5	-0.628008	1.277725	23	24

```

6  0.003116  2.125808  19      35
7 -0.281785  0.864108  24      16
8 -1.018384  1.563524  37      18
9 -1.709348  0.952062  30      21

```

Se visualiza la información en dos dimensiones, para analizar si se separan de alguna forma clara las clases de hombres internos por su edad, sentencia o si no hay unión entre estos dos factores.

```

In [11]: import ssl
import pandas as pd
from numpy import isnan, nan
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv('practica7.csv')

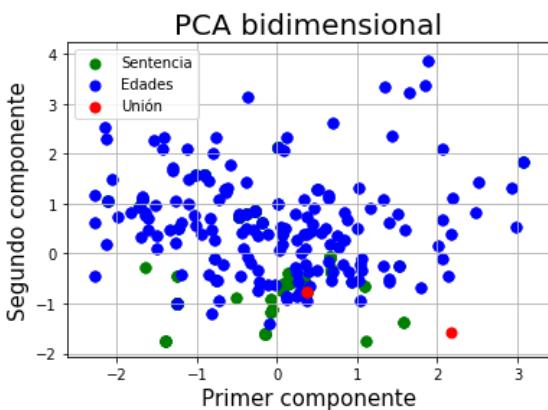
pri = d.sentenceYears
seg = d.age >=16
ter = ~ (pri | seg)

plt.title('PCA bidimensional', fontsize = 20)
plt.xlabel('Primer componente', fontsize = 15)
plt.ylabel('Segundo componente', fontsize = 15)
plt.scatter(d.loc[pri].comp_0, d.loc[pri].comp_1, c = 'g', s = 50)
plt.scatter(d.loc[seg].comp_0, d.loc[seg].comp_1, c = 'b', s = 50)
plt.scatter(d.loc[ter].comp_0, d.loc[ter].comp_1, c = 'r', s = 50)
plt.legend(['Sentencia', 'Edades', 'Unión'])
plt.grid()

C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:22: FutureWarning:
Passing list-like to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-reindex-listlike

```



Con el diagrama anterior se ve que se separan un poco el factor edad de los hombres con el factor años de sentencia que tienen, en cambio la unión de estos factores es casi nula. Entonces en este caso y con este análisis se pudieran separar de forma clara estos factores.

```
In [ ]:
```

1. Discución de la Práctica 9

La **Práctica 9** consiste en realizar pronósticos con statsmodels. Se realizan pronósticos para predecir el valor que tomará una variable a partir de valores que haya tomado anteriormente. Se trabaja en pronosticar la cantidad de años de sentencia que se les imponen a los internos basándose en algunas de las características personales y económicas de los internos.

La tarea fue entregada en tiempo y calificada con calificación 2 ya que por falta de tiempo no le pude dedicar todo el esfuerzo que conllevaba hacer esta práctica y no entendí bien lo que me pedía la profesora en esta práctica.



REPORTE PRÁCTICA 9: Pronósticos con statsmodels

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

En esta práctica se realizan pronósticos para predecir el valor que tomará una variable a partir de valores que haya tomado anteriormente.

El pronóstico se puede decir que es la predicción de lo que sucederá con un elemento determinado dentro del marco de un conjunto dado de condiciones. El pronóstico es muy útil en cualquiera de las ramas en que se utilice, por ejemplo, en el plano empresarial ayuda a reducir el rango de incertidumbre dentro del cual se toman las decisiones que afectan el futuro del negocio, que aunque el pronóstico no sustituye el juicio administrativo en la toma de decisiones, si ayuda en ese proceso.

Se va a intentar pronosticar la cantidad de años de sentencia que se les imponen a los internos basándose en las características personales y económicas de los internos:

- el nivel educacional
- la edad
- si tienen hijos
- su estado civil
- el ingreso mensual que tenían antes de estar privados a la libertad.

Para esto se necesita normalizar alguna de estas características para que estén entre cero y uno. La normalización aplicada en este caso sigue la siguiente fórmula:

$$\text{normalización} = (x - \min(x)) / (\max(x) - \min(x))$$

```
In [105]: import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from numpy import isnan
import pandas as pd
import ssl

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6b.csv")

o.columns = ['age', 'education', 'prevRelStatus', 'incomePrev', 'hasChildren', 'sentenceYears']

o['age'] = o['age'].fillna(0)
o['education'] = o['education'].fillna(0)
o['prevRelStatus'] = o['prevRelStatus'].fillna(0)
o['incomePrev'] = o['incomePrev'].fillna(0)
o['hasChildren'] = o['hasChildren'].fillna(0)
o['sentenceYears'] = o['sentenceYears'].fillna(0)

x1 = pd.to_numeric(o.age)
x2 = pd.to_numeric(o.education)
x3 = pd.to_numeric(o.prevRelStatus)
x4 = pd.to_numeric(o.incomePrev)
x5 = pd.to_numeric(o.hasChildren)
x6 = pd.to_numeric(o.sentenceYears)

df = pd.DataFrame({"Edad": x1, "Edu": x2, "EC": x3, "IM": x4, "TH": x5, "Sent": x6})

scaler = MinMaxScaler()
scaled_df = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_df, columns=['Edad', 'Edu', 'EC', 'IM', 'TH', 'Sent'])
print(scaled_df.iloc[:10, :10])

scaled_df.to_csv('practica8.dat')
```

	Edad	Edu	EC	IM	TH	Sent
0	0.065217	0.285714	0.282828	1.0	0.50	0.0
1	0.434783	0.142857	0.353535	0.5	0.25	1.0
2	0.391304	0.142857	0.393939	0.5	0.75	0.2
3	0.086957	0.142857	0.323232	0.5	1.00	0.0
4	0.086957	0.428571	0.252525	0.5	0.50	1.0
5	0.304348	0.285714	0.404040	0.5	0.50	0.0
6	0.489130	0.285714	0.363636	1.0	1.00	0.0
7	0.271739	0.142857	0.000000	0.5	1.00	0.4
8	0.184783	0.285714	0.414141	1.0	0.50	1.0
9	0.206522	0.285714	0.545455	1.0	0.25	0.0

Ya teniendo todas las columnas normalizadas y con valores numéricos se realiza un primer pronóstico con el método **Holt**.

```
In [111]: from statsmodels.tsa.api import Holt
import matplotlib.pyplot as plt
from numpy import asarray
import pandas as pd
import ssl

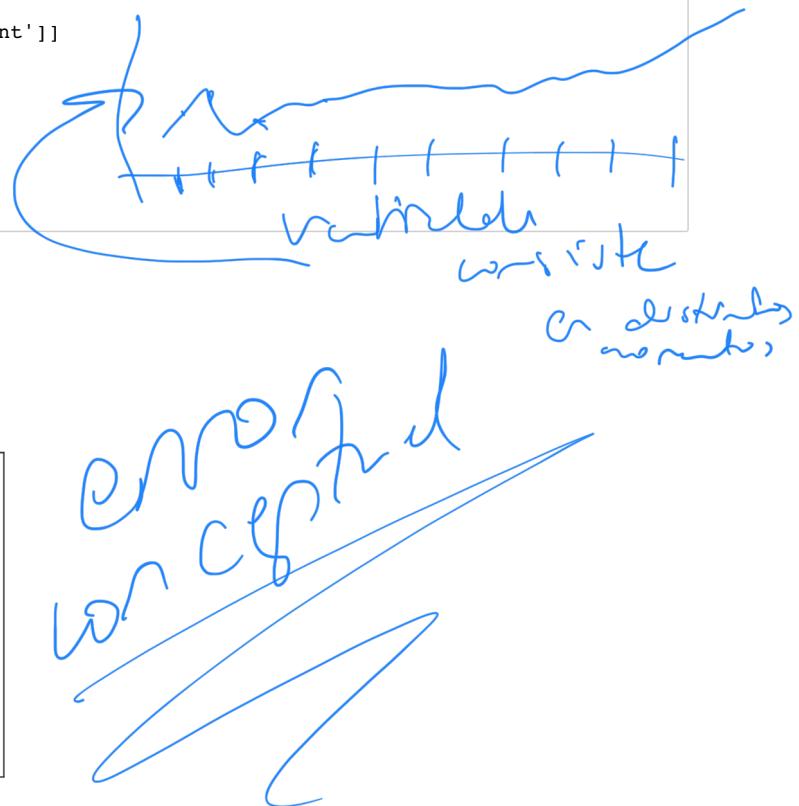
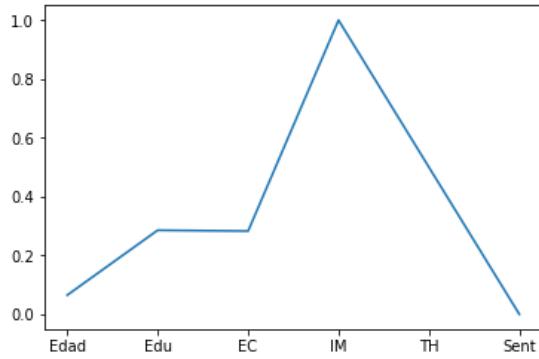
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica8.dat")

o.columns = ['', 'Edad', 'Edu', 'EC', 'IM', 'TH', 'Sent']

o = o[['Edad', 'Edu', 'EC', 'IM', 'TH', 'Sent']]

ejemplo = o.loc[0,:]
print(ejemplo)
x = range(len(ejemplo))
plt.plot(x, asarray(ejemplo))
plt.xticks(x, ejemplo.index)
plt.show()

Edad      0.065217
Edu       0.285714
EC        0.282828
IM        1.000000
TH        0.500000
Sent      0.000000
Name: 0, dtype: float64
```



La gráfica no muestra ninguna estacionalidad o tendencia, lo que sugiere que el conjunto de datos diferenciado por temporada es un buen punto de partida para el modelado.

```
In [112]: from statsmodels.tsa.api import Holt
import matplotlib.pyplot as plt
from numpy import asarray
import pandas as pd
import ssl

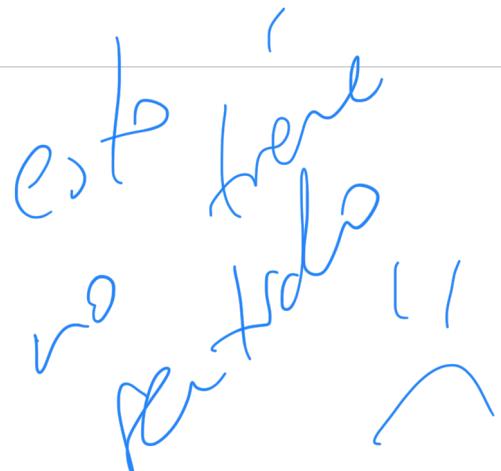
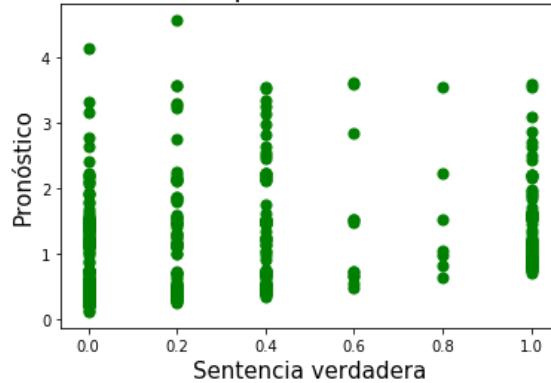
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv("practica8.dat")

d.columns = ['', 'Edad', 'Edu', 'EC', 'IM', 'TH', 'Sent']

d = d[['Edad', 'Edu', 'EC', 'IM', 'TH', 'Sent']]

lbls = d.columns
x = range(len(lbls))
pronosticos = []
for i in range(len(d)):
    y = asarray(d.loc[i,:])
    f = Holt(asarray(y)).fit(smoothing_level = 0.1)
    pronosticos.append(f.forecast(1))
plt.title('Pronóstico de un paso con el método de Holt', fontsize = 20)
plt.xlabel('Sentencia verdadera', fontsize = 15)
plt.ylabel('Pronóstico', fontsize = 15)
plt.scatter(d.Sent, pronosticos, c = 'g', s = 50)
plt.show()
```

Pronóstico de un paso con el método de Holt



Lo anterior a simple vista muestra que están bien clasificados, por lo que debería de ser un buen pronóstico. Se usa este conjunto de datos como una entrada para el modelo ARIMA (<https://machinelearningmastery.com/time-series-forecast-study-python-monthly-sales-french-champagne/>).

```
In [139]: import matplotlib.pyplot as plt
from numpy import asarray
import pandas as pd
import ssl

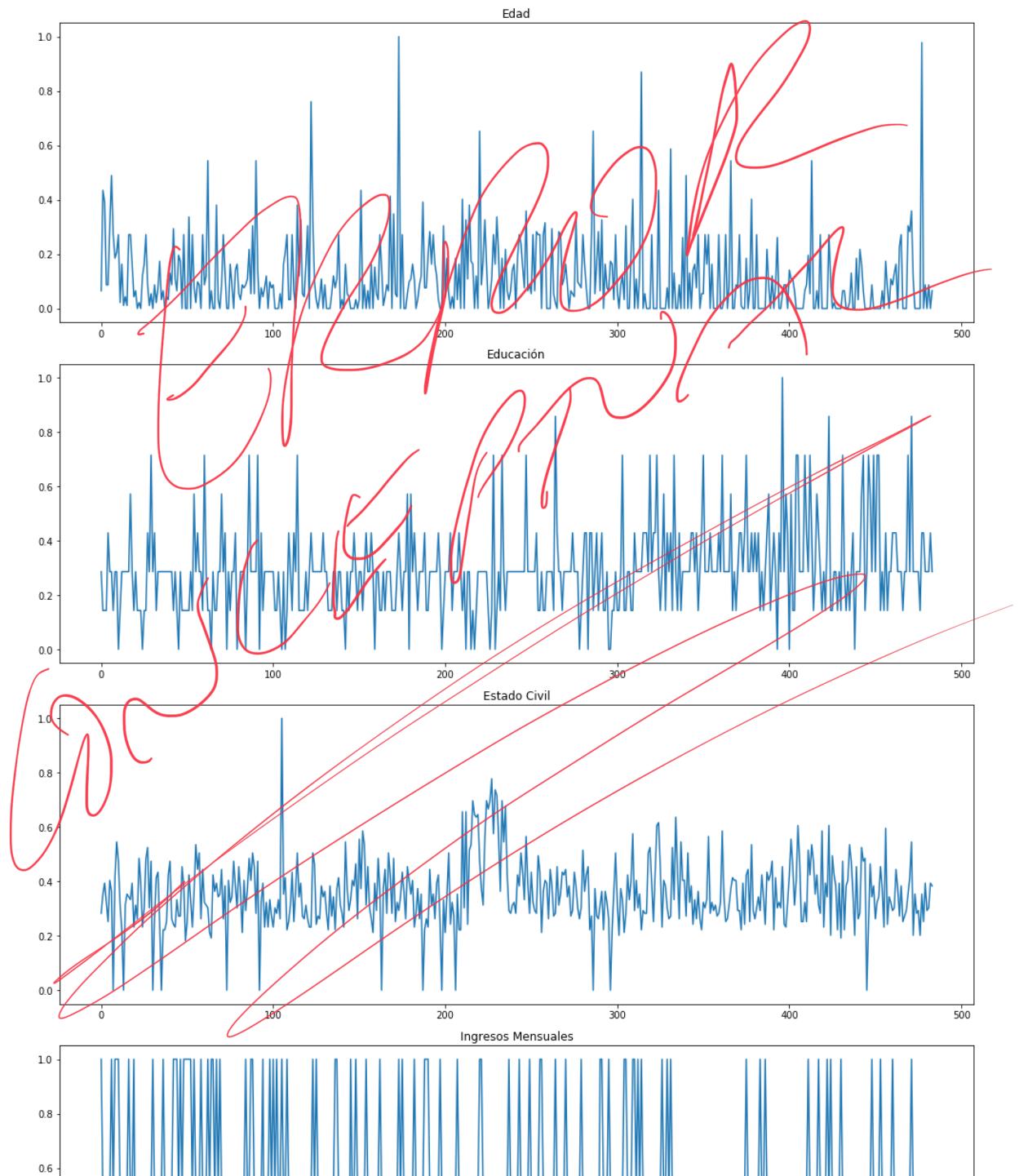
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv('practica8.dat')
x = range(len(d))

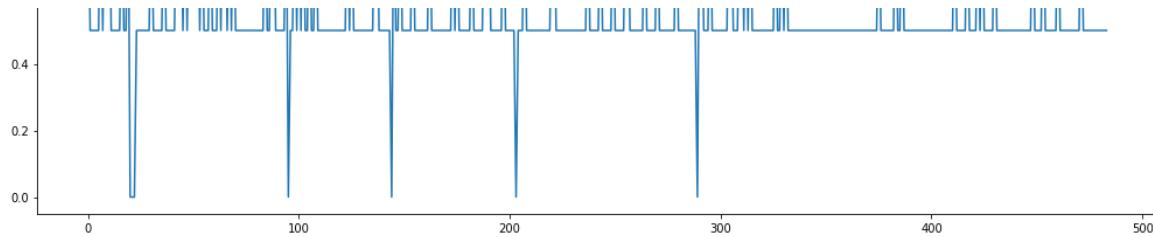
plt.rcParams["figure.figsize"] = [14, 20]
f = plt.figure()
sf = f.add_subplot(411)
y = asarray(d['Edad'])
sf.plot(x, y)
sf.set_title('Edad')
sf = f.add_subplot(412)
y = asarray(d['Edu'])
```

```

sf.set_title('Educación')
sf.plot(x, y)
sf = f.add_subplot(413)
y = asarray(d['EC'])
sf.set_title('Estado Civil')
sf.plot(x, y)
sf = f.add_subplot(414)
y = asarray(d['IM'])
sf.set_title('Ingresos Mensuales')
sf.plot(x, y)
plt.tight_layout()
plt.show()

```





Se sigue probando que la serie sea estacionaria con el siguiente código.

```
In [146]: from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
import pandas as pd
import ssl

def test_stationarity(ts, w, r, i):
    rolmean = ts.rolling(w).mean()
    rolstd = ts.rolling(w).std()
    plt.subplot(r, 1, i)
    orig = plt.plot(ts, color='blue', label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc = 'best')
    plt.title('Rolling Mean & Standard Deviation')
    dfoutput = adfuller(ts, autolag='AIC')
    dfoutput = pd.Series(dfoutput[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dfoutput[4].items():
        dfoutput['Critical Value {:.3f}'.format(key)] = value
    return 'Results of Dickey-Fuller Test:\n' + '\n'.join(['{:s}\t{:,.3f}'.format(k, v) for (k, v) in dfoutput.items()])

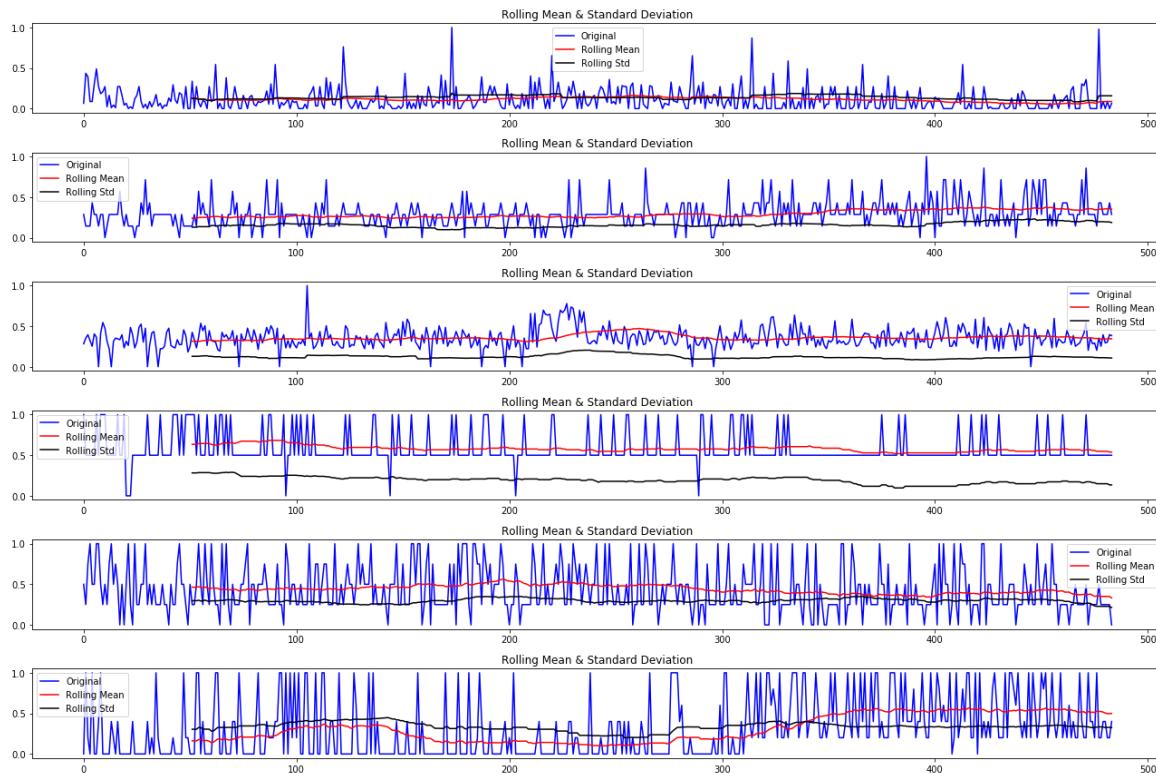
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv('practica8.dat')
plt.rcParams["figure.figsize"] = [18, 12]
f = plt.figure()
i = 1
lvls = ['Edad', 'Edu', 'EC', 'IM', 'TH', 'Sent']
r = len(lvls)
t = ''
w = 52
for c in lvls:
    t += test_stationarity(d[c], w, r, i)
    i += 1
plt.tight_layout()
print(t)

Results of Dickey-Fuller Test:
Test Statistic -23.133
p-value 0.000
#Lags Used 0.000
Number of Observations Used 483.000
Critical Value (1%) -3.444
Critical Value (5%) -2.868
Critical Value (10%) -2.570Results of Dickey-Fuller Test:
Test Statistic -11.020
p-value 0.000
#Lags Used 2.000
Number of Observations Used 481.000
Critical Value (1%) -3.444
Critical Value (5%) -2.868
Critical Value (10%) -2.570Results of Dickey-Fuller Test:
Test Statistic -4.707
p-value 0.000
#Lags Used 8.000
```

```

Number of Observations Used      475.000
Critical Value (1%)      -3.444
Critical Value (5%)       -2.868
Critical Value (10%)      -2.570Results of Dickey-Fuller Test:
Test Statistic   -13.679
p-value 0.000
#Lags Used      1.000
Number of Observations Used      482.000
Critical Value (1%)      -3.444
Critical Value (5%)       -2.868
Critical Value (10%)      -2.570Results of Dickey-Fuller Test:
Test Statistic   -9.909
p-value 0.000
#Lags Used      3.000
Number of Observations Used      480.000
Critical Value (1%)      -3.444
Critical Value (5%)       -2.868
Critical Value (10%)      -2.570Results of Dickey-Fuller Test:
Test Statistic   -3.118
p-value 0.025
#Lags Used      10.000
Number of Observations Used      473.000
Critical Value (1%)      -3.444
Critical Value (5%)       -2.868
Critical Value (10%)      -2.570

```



En lo anterior, los valores P están por debajo de cualquier nivel de significancia razonable, por lo cual se evidencia que no existe tendencia. Luego se comprueba que tan bien se pronosticó en esta práctica con el siguiente código, comparando los resultados verdaderos con los del pronóstico. Se utilizan dos lógicas para comprobar y se evidencian en las gráficas que se explican luego.

```
In [153]: from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error
```

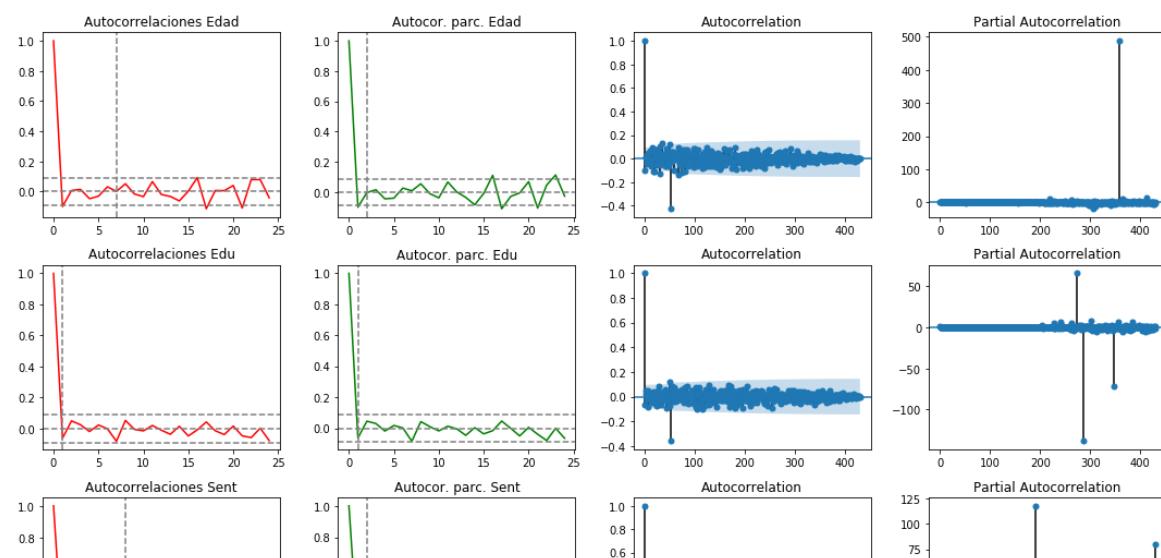
```

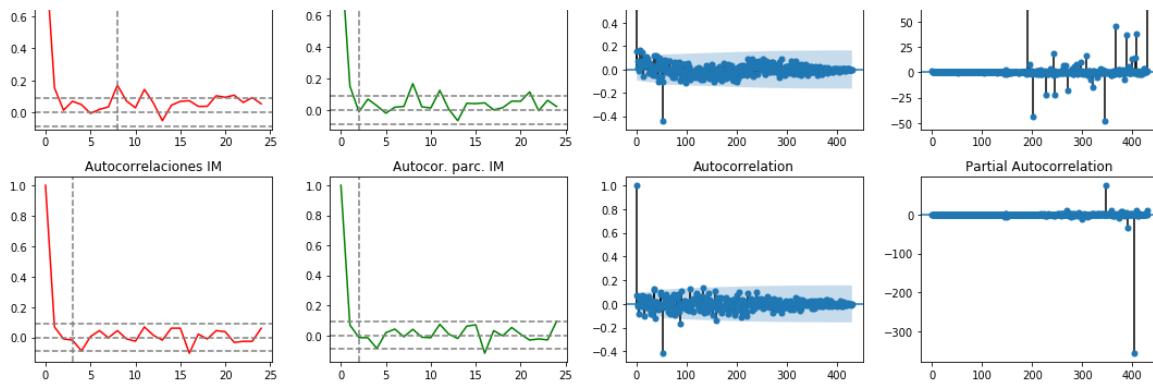
from math import sqrt
import ssl

def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return pd.Series(diff)

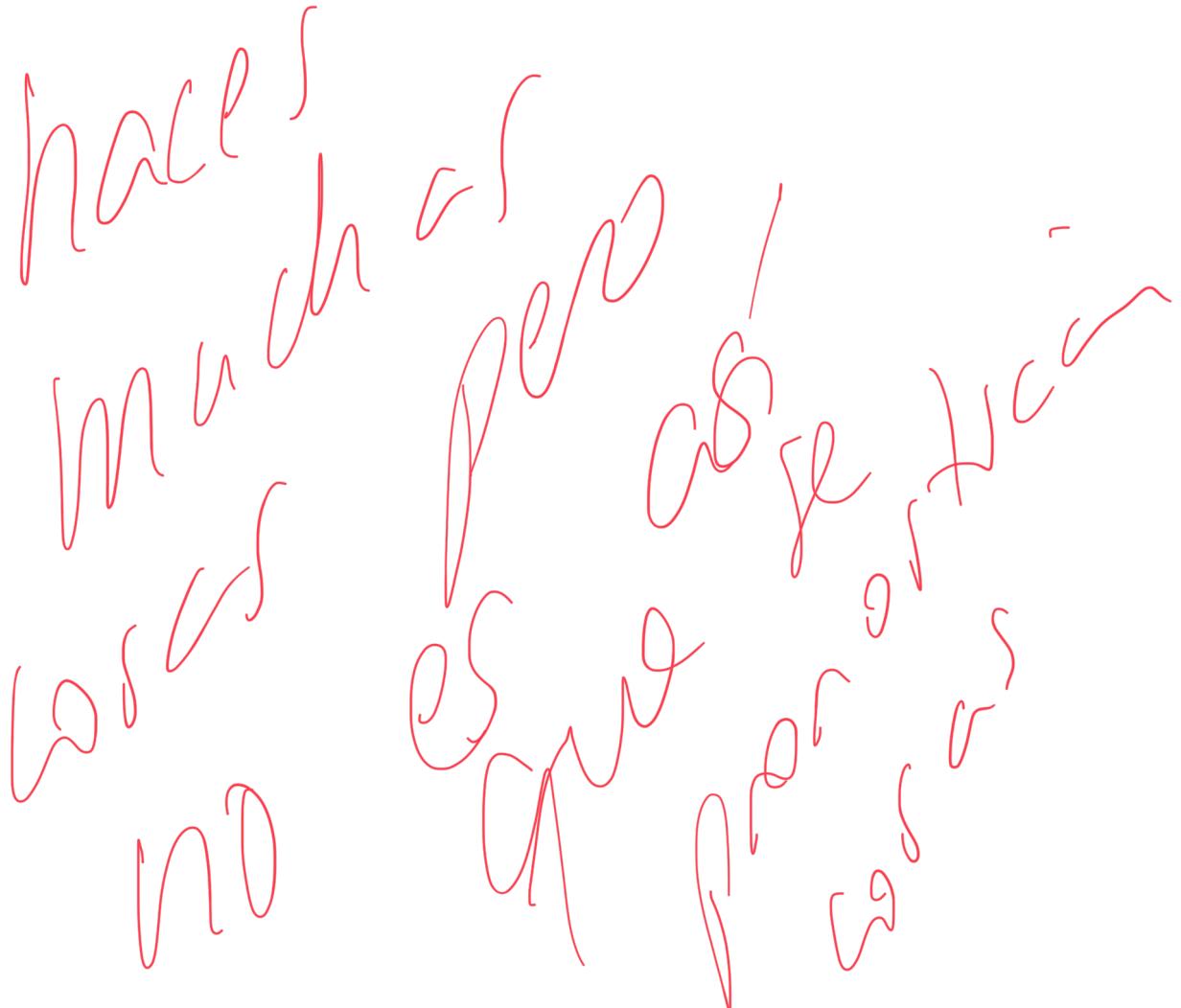
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv('practica8.dat')
d = d.iloc[:-1, :]
plt.rcParams["figure.figsize"] = [15, 12]
f = plt.figure()
i = 1
n = len(d)
q = {'Edad': 7, 'Edu': 1, 'Sent': 8, 'IM': 3}
p = {'Edad': 2, 'Edu': 1, 'Sent': 2, 'IM': 2}
for c in ['Edad', 'Edu', 'Sent', 'IM']:
    ts = difference(asarray(d[c]), interval = 52).dropna()
    # primera logica
    plt.subplot(4, 4, i)
    i += 1
    plt.plot(acf(ts, nlags = 24), 'r')
    plt.axhline(y = 0, linestyle='--', color = 'gray')
    plt.axvline(x = q[c], linestyle = '--', color='gray')
    plt.axhline(y = -1.96 / sqrt(n), linestyle = '--', color='gray')
    plt.axhline(y = 1.96 / sqrt(n), linestyle = '--', color='gray')
    plt.title('Autocorrelaciones ' + c)
    plt.subplot(4, 4, i)
    i += 1
    plt.plot(pacf(ts, nlags = 24, method='ols'), 'g')
    plt.axhline(y = 0, linestyle = '--', color = 'gray')
    plt.axvline(x = p[c], linestyle = '--', color='gray')
    plt.axhline(y = -1.96 / sqrt(n),linestyle = '--', color='gray')
    plt.axhline(y = 1.96 / sqrt(n),linestyle = '--', color='gray')
    plt.title('Autocor. parc. ' + c)
    # segunda logica
    plt.subplot(4, 4, i)
    i += 1
    plot_acf(ts, ax = plt.gca())
    plt.subplot(4, 4, i)
    i += 1
    plot_pacf(ts, ax = plt.gca())
plt.tight_layout()

```





Las primeras dos columnas de gráficos corresponden a la lógica uno utilizada para comprobar que tan bueno es el pronóstico, buscando la coordenada x del primer cruce con el intervalo de confianza superior, se obtiene $q = \{\text{'Edad': 7, 'Edu': 1, 'Sent': 8, 'IM': 3}\}$ $p = \{\text{'Edad': 2, 'Edu': 1, 'Sent': 2, 'IM': 2}\}$. Las otras dos columnas que le siguen corresponden a la segunda lógica utilizada, donde se ven mal todas las autocorrelaciones y además se muestran picos en las parciales que no son buena señal.



```
In [155]: from statsmodels.tsa.arima_model import ARIMA
import pandas as pd
import ssl

# https://machinelearningmastery.com/time-series-forecast-study-python-monthly-sales-french-champagne/
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return pd.Series(diff)

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
d = pd.read_csv('practica8.dat')
guardar = d.iloc[-1:, :]
d = d.iloc[:-1, :]
n = len(d)
q = {'Edad': 7, 'Edu': 1, 'Sent': 8, 'IM': 3}
p = {'Edad': 2, 'Edu': 1, 'Sent': 2, 'IM': 2}
prons = []
for c in ['Edad', 'Edu', 'Sent', 'IM']:
    ts = difference(asarray(d[c])), interval = 52).dropna()
    try:
        m = ARIMA(ts, order = (p[c], 1, q[c]))
        f = m.fit(trend = 'nc', disp = 0)
        print('{:s}\tpron.\t'.format(c), int(round(f.forecast()[0][0] + d[c].iloc[n - 52])))
        print('{:s}\treal\t'.format(c), guardar[c].iloc[0])
    except:
        print(c, "no se pudo pronosticar con esos parámetros")
    print('...')

Edad no se pudo pronosticar con esos parámetros
...
Edu      pron.      1
Edu      real     0.2857142857142857
...
Sent     pron.      0
Sent     real     0.4
...
IM       pron.      0
IM       real     0.5
...
...
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:488: HessianInversionWarning: Inverting hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
```

La edad de los internos no se pudo pronosticar, en el caso de los demás datos no dan pronósticos buenos, pero si se pueden pronosticar, incluyendo la sentencia de los internos, partiendo de las características personales y económicas de estos.



1. Discución de la Práctica 10

La **Práctica 10** consiste en realizar clasificaciones de datos con sklearn. El objetivo de esta práctica consiste en utilizar algunos métodos de clasificación a por lo menor una división de interés en los datos de los internos. Se realiza además una proyección o selección bidimensional de atributos para poder visualizar el resultado.

La tarea fue entregada en tiempo y calificada con calificación 6 penalizando un punto ya que la profesora estima conveniente mejorar la claridad de las explicaciones y justificar mejor los resultados obtenidos.

REPORTE PRÁCTICA 10: Clasificación de datos con sklearn

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio que permitirá evaluar las condiciones de reinserción social dentro del CERESO "Apodaca" y dentro del Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

El siguiente reporte se trabaja utilizando algunos métodos de clasificación a por lo menor una división de interés en los datos de los internos. Se realiza una proyección o selección bidimensional de atributos para poder visualizar el resultado.

Primeramente, se instalan los paquetes necesarios realizar este trabajo, luego se analizan los factores que resultaron significativos en su correlación en prácticas anteriores, como los ingresos mensuales que tenían los internos, para los que se utiliza la nomenclatura siguiente:

1. Menos de 3,000
2. De 3,001 a 6,000
3. De 6,001 a 9,000
4. Más de 9,000

Se usa para el clasificador, los ingresos por encima de 9000 pesos, con la etiqueta 4. *poco daño*

Se tiene en cuenta para el clasificador la edad de los internos, el cual fue otro de los factores de mayor significancia en prácticas anteriores. Se tiene en cuenta las edades mayores a 16 años ya que en algunas respuestas a esta pregunta los internos respondían edades menores a 16 y no es verdadero.

Finalmente, los valores que se colocan las etiquetas con las que se va a realizar la clasificación son:

- Internos con edad mayor a 16
- Ingresos mensuales mayores a 9000 (con la etiqueta 4)
- Los que no cumplen con las condiciones anteriores

? *poco daño*

```
In [117]: import ssl  
import pandas as pd  
from math import ceil, sqrt  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA  
from matplotlib.colors import ListedColormap  
from numpy import isnan, nan, arange, meshgrid, c_  
from sklearn.model_selection import train_test_split
```

```

from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB


if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
d = pd.concat([o.loc[o.age >= 16], o.loc[~isnan(o.age)]])


cat = pd.Categorical(d.crimen)
d.crimen = cat.codes
cat2 = pd.Categorical(d.educacion)
d.educacion = cat2.codes

pri = d.incomePrev >= 4
seg = d.age >= 16
ter = ~pri & ~seg

```

? NO shool now

```

d['etiquetas'] = [1 if pri[i].all() else (2 if seg[i].all() else (0 if ter[i].all() else "NA")))
for i in pri.keys() # etiquetas
print(d.etiquetas.value_counts())
y = d.etiquetas

xVars = ['sentenceYears', 'crimen', 'educacion']

x = d.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
pca = PCA(n_components = 2)
X = pca.fit_transform(x)

h = 0.02 #tamaño de paso en la malla

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
               SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
               DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max \
               _features=1), \
               AdaBoostClassifier(), GaussianNB()]
k = int(ceil(sqrt(len(classifiers) + 1)))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42) # división
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = meshgrid(arange(x_min, x_max, h), arange(y_min, y_max, 0.02))
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
plt.rcParams["figure.figsize"] = [16, 16]
figure = plt.figure()
ax = plt.subplot(k, k, 1)
ax.set_title("Datos de entrada")
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, alpha=0.2, edgecolors='k') # entrenamiento
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.2, edgecolors='k') # validación
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())

```

```

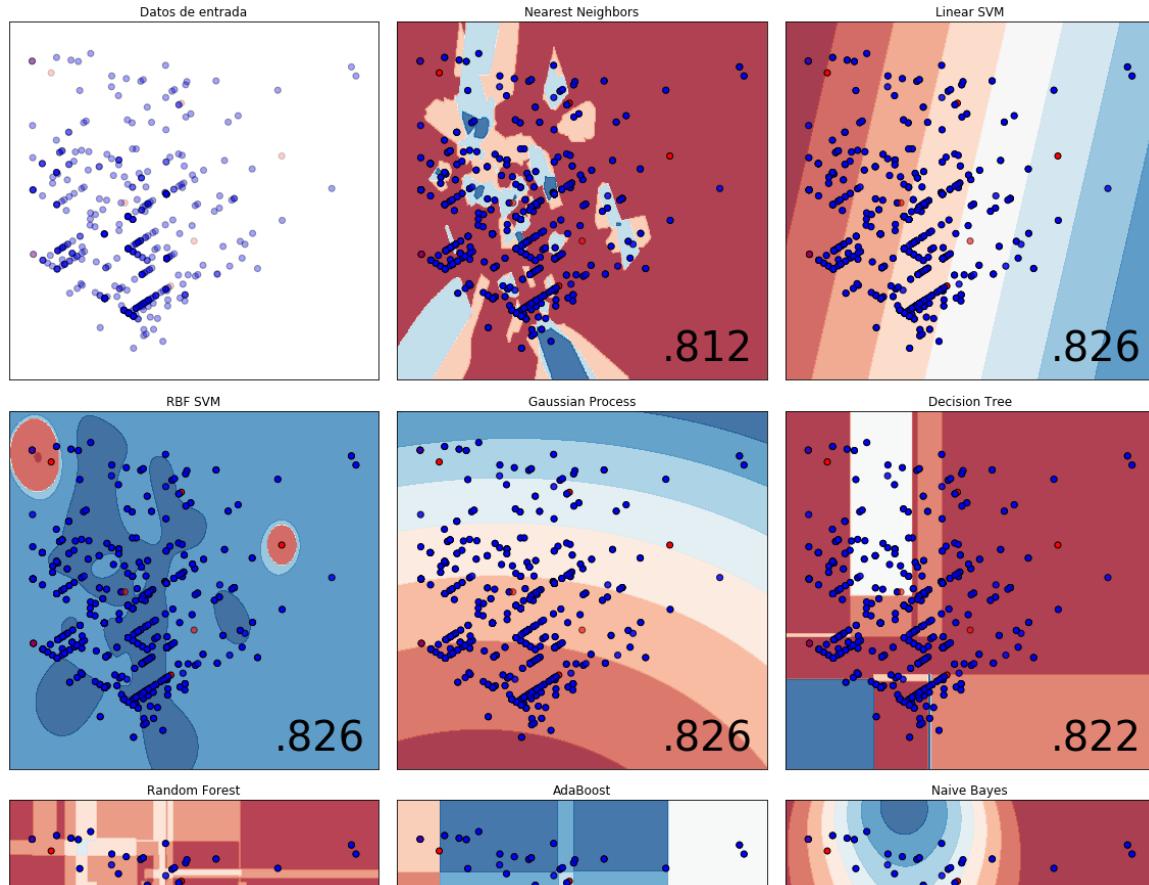
i = 2
for name, clf in zip(names, classifiers):
    ax = plt.subplot(k, k, i)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    if hasattr(clf, "decision_function"):
        Z = clf.decision_function(c_[xx.ravel(), yy.ravel()])[:, 1]
    else:
        Z = clf.predict_proba(c_[xx.ravel(), yy.ravel()])[:, 1]
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)
    ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors='k')
    ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, edgecolors='k', alpha=0.6)
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(name)
    ax.text(xx.max() - .3, yy.min() + .3, ('%.3f' % score).lstrip('0'), size=40, horizontalalignment='right')
    i += 1
plt.tight_layout()
plt.show()
2     808
1     136
0      11
Name: etiquetas, dtype: int64

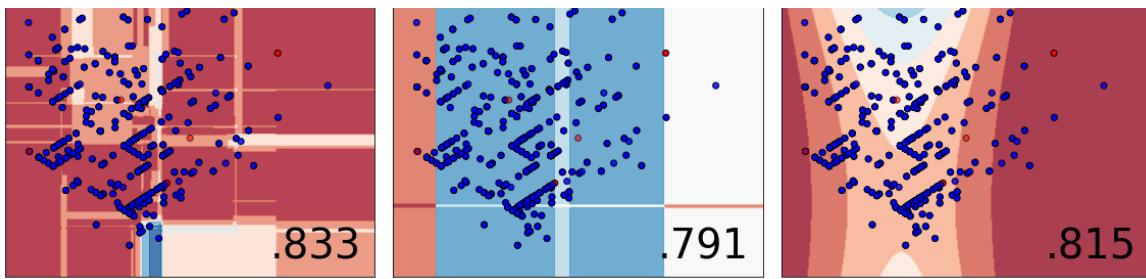
```

```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
    warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
    warnings.warn(msg, DataConversionWarning)

```





La parte inferior derecha de las imágenes muestra la precisión de la clasificación en el conjunto de prueba. Para poder conocer cuántos datos fueron clasificados correctamente y cuáles eran los errores, se utiliza la [matrices de confusión](https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py) (https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py), además se podrá comprobar si fueron buenos los modelos utilizados.

```
In [118]: import ssl
import pandas as pd
from math import ceil, sqrt
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
from numpy import isnan, nan
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
d = pd.concat([o.loc[o.age >= 16], o.loc[~isnan(o.age)]])

cat = pd.Categorical(d.crimen)
d.crimen = cat.codes
cat2 = pd.Categorical(d.educacion)
d.educacion = cat2.codes

pri = d.incomePrev >= 4
seg = d.age >= 16
ter = ~pri & ~seg

d['etiquetas'] = [1 if pri[i].all() else (2 if seg[i].all() else (0 if ter[i].all() else "NA")))
for i in pri.keys()] # etiquetas
print(d.etiquetas.value_counts())
y = d.etiquetas

xVars = ['sentenceYears', 'crimen', 'educacion']

x = d.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
pca = PCA(n_components = 2)
X = pca.fit_transform(x)

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
               SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
               DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max
```

```

    _features=1), \
        AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42) # la misma división
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name, clf.score(X_test, y_test))
    expected, predicted = y_test, clf.predict(X_test)
    print(metrics.classification_report(expected, predicted))
    print(metrics.confusion_matrix(expected, predicted))
    print('-' * 60)

2      808
1     136
0      11
Name: etiquetas, dtype: int64
Nearest Neighbors 0.8083623693379791
      precision    recall   f1-score   support
          0       0.50     0.25     0.33       4
          1       0.41     0.28     0.33      46
          2       0.86     0.92     0.89     237
   micro avg       0.81     0.81     0.81     287
   macro avg       0.59     0.48     0.52     287
weighted avg       0.78     0.81     0.79     287

[[ 1  1  2]
 [ 0 13 33]
 [ 1 18 218]]
-----
Linear SVM 0.8257839721254355
      precision    recall   f1-score   support
          0       0.00     0.00     0.00       4
          1       0.00     0.00     0.00      46
          2       0.83     1.00     0.90     237
   micro avg       0.83     0.83     0.83     287
   macro avg       0.28     0.33     0.30     287
weighted avg       0.68     0.83     0.75     287

[[ 0  0  4]
 [ 0  0 46]
 [ 0  0 237]]
-----
RBF SVM 0.8257839721254355
      precision    recall   f1-score   support
          0       0.00     0.00     0.00       4
          1       0.00     0.00     0.00      46
          2       0.83     1.00     0.90     237
   micro avg       0.83     0.83     0.83     287
   macro avg       0.28     0.33     0.30     287
weighted avg       0.68     0.83     0.75     287

[[ 0  0  4]
 [ 0  0 46]
 [ 0  0 237]]
-----
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no
  ...

```

```

predicted samples.
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
'precision', 'predicted', average, warn_for)

Gaussian Process 0.8257839721254355
      precision    recall   f1-score   support
          0         0.00     0.00     0.00       4
          1         0.00     0.00     0.00      46
          2         0.83     1.00     0.90     237
micro avg       0.83     0.83     0.83     287
macro avg       0.28     0.33     0.30     287
weighted avg     0.68     0.83     0.75     287

[[ 0  0  4]
 [ 0  0  46]
 [ 0  0 237]]


Decision Tree 0.8222996515679443
      precision    recall   f1-score   support
          0         0.00     0.00     0.00       4
          1         0.33     0.02     0.04      46
          2         0.83     0.99     0.90     237
micro avg       0.82     0.82     0.82     287
macro avg       0.39     0.34     0.31     287
weighted avg     0.74     0.82     0.75     287

[[ 0  0  4]
 [ 0  1  45]
 [ 0  2 235]]


Random Forest
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
'precision', 'predicted', average, warn_for)

0.8257839721254355
      precision    recall   f1-score   support
          0         0.00     0.00     0.00       4
          1         0.00     0.00     0.00      46
          2         0.83     1.00     0.90     237
micro avg       0.83     0.83     0.83     287
macro avg       0.28     0.33     0.30     287
weighted avg     0.68     0.83     0.75     287

[[ 0  0  4]
 [ 0  0  46]
 [ 0  0 237]]


AdaBoost 0.7909407665505227
      precision    recall   f1-score   support

```

```

          0      0.14      0.25      0.18      4
          1      0.11      0.02      0.04     46
          2      0.83      0.95      0.89    237

  micro avg      0.79      0.79      0.79    287
  macro avg      0.36      0.41      0.37    287
weighted avg      0.71      0.79      0.74    287

[[ 1  0  3]
 [ 2  1  43]
 [ 4  8 225]]
-----
Naive Bayes 0.8153310104529616
      precision    recall   f1-score   support
          0       0.00     0.00     0.00      4
          1       0.00     0.00     0.00     46
          2       0.82     0.99     0.90    237

  micro avg      0.82      0.82      0.82    287
  macro avg      0.27      0.33      0.30    287
weighted avg      0.68      0.82      0.74    287

[[ 0  0  4]
 [ 0  0  46]
 [ 3  0 234]]
-----
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)

```

Los elementos diagonales representan el número de puntos para los cuales la etiqueta predicha es igual a la etiqueta verdadera, mientras que los elementos fuera de la diagonal son aquellos que están clasificados incorrectamente por el clasificador. Cuanto más altos sean los valores diagonales de la matriz de confusión, mejor, lo que indica muchas predicciones correctas.

En la diagonal se encuentran los valores de mayor tamaño, y en varias clasificaciones se llega a tener 0 errores, pero no la perfección de la matriz, por lo cual se intenta probar con una sola etiqueta, la de ingresos mensuales y se aumenta a 0.4 el porcentaje de prueba.

```
In [119]: import ssl
import pandas as pd
from math import ceil, sqrt
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
from numpy import isnan, nan, arange, meshgrid, c_
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB


if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
d = pd.concat([o.loc[o.age >= 16], o.loc[~isnan(o.age)]])


cat = pd.Categorical(d.crimen)
```

```

d.crimen = cat.codes
cat2 = pd.Categorical(d.educacion)
d.educacion = cat2.codes

pri = d.incomePrev >= 4

d['etiquetas'] = [1 if pri[i].all() else 0 for i in pri.keys()]
print(d.etiquetas.value_counts())
y = d.etiquetas

xVars = ['sentenceYears', 'crimen', 'educacion']

x = d.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
pca = PCA(n_components = 2)
X = pca.fit_transform(x)

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
               SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
               DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max_\
               _features=1), \
               AdaBoostClassifier(), GaussianNB()]
k = int(ceil(sqrt(len(classifiers) + 1)))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=42)
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = meshgrid(arange(x_min, x_max, h), arange(y_min, y_max, 0.02))
cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
plt.rcParams["figure.figsize"] = [16, 16]
figure = plt.figure()
ax = plt.subplot(k, k, 1)
ax.set_title("Datos de entrada")
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, alpha=0.2, edgecolors='k')
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, alpha=0.2, edgecolors='k')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
i = 2
for name, clf in zip(names, classifiers):
    ax = plt.subplot(k, k, i)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    if hasattr(clf, "decision_function"):
        Z = clf.decision_function(c_[xx.ravel(), yy.ravel()])
    else:
        Z = clf.predict_proba(c_[xx.ravel(), yy.ravel()])[:, 1]
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)
    ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright, edgecolors='k')
    ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm_bright, edgecolors='k', alpha=0.6)
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(name)
    ax.text(xx.max() - .3, yy.min() + .3, ('%.3f' % score).lstrip('0'), size=40, horizontalalignment='right')
    i += 1
plt.tight_layout()
plt.show()

```

```

0      819
1      136
Name: etiquetas, dtype: int64

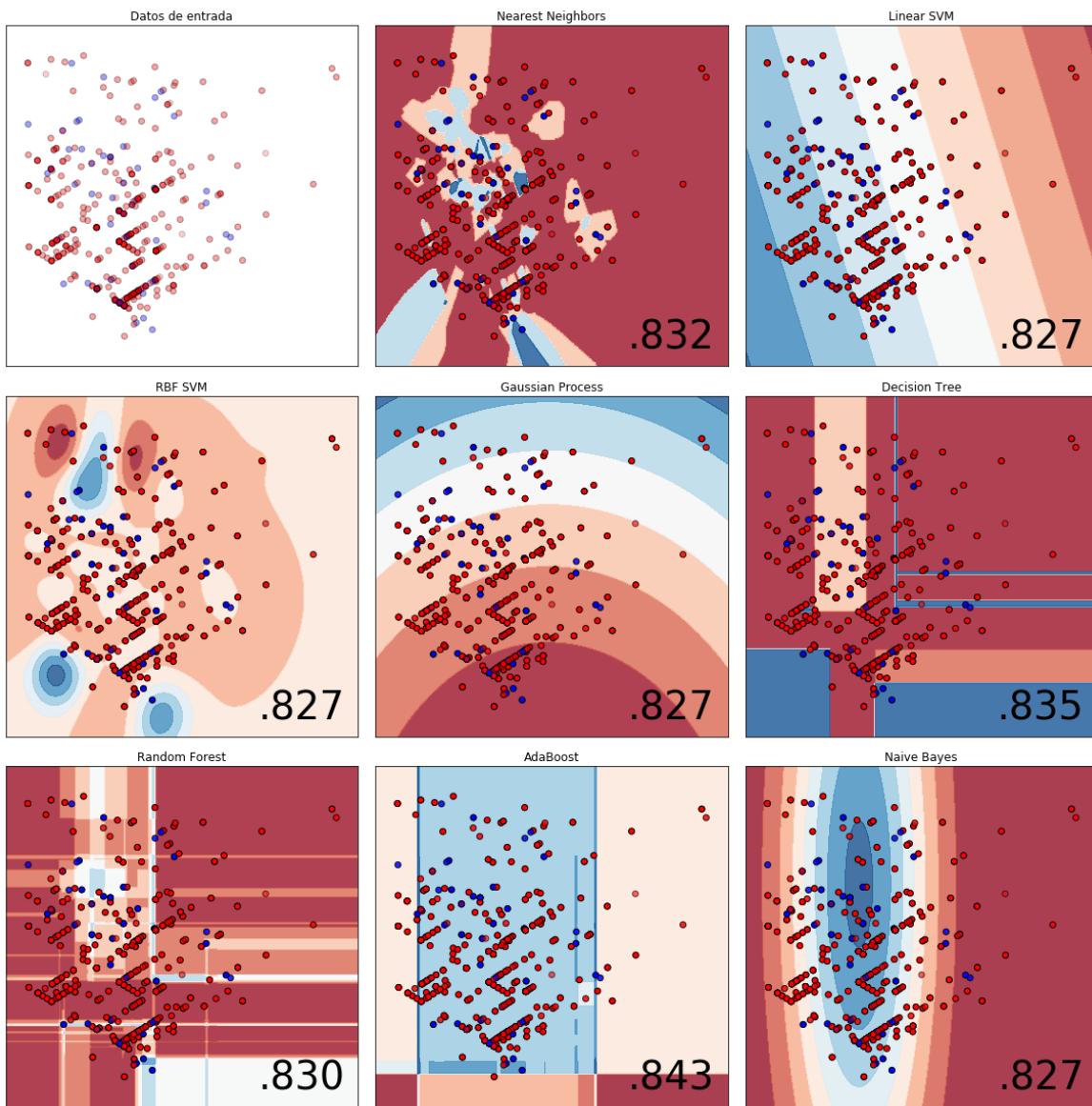
```

```

C:\Users\Diego\Downloads\146\mifit\analisis\multivariado\validation\20150519\ReportePractica10_Evely.ipynb

```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning
  warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning
  warnings.warn(msg, DataConversionWarning)
```



Los valores de las esquinas inferiores aumentaron un poco, veamos las matrices de confusión para ver qué tan buena es la clasificación anterior.

```
In [113]: import ssl
import pandas as pd
from math import ceil, sqrt
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
from numpy import isnan, nan
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

```

from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
d = pd.concat([o.loc[o.age >= 16], o.loc[~isnan(o.age)]])
d = d.drop(['Unnamed: 0', 'Unnamed: 0.1', 'hasChildren'], 1)

cat = pd.Categorical(d.crimen)
d.crimen = cat.codes
cat2 = pd.Categorical(d.educacion)
d.educacion = cat2.codes
cat3 = pd.Categorical(d.gender)
d.gender = cat3.codes

pri = d.incomePrev >= 4

d['etiquetas'] = [1 if pri[i].all() else 0 for i in pri.keys()]
print(d.etiquetas.value_counts())
y = d.etiquetas

xVars = ['sentenceYears', 'crimen', 'educacion']

x = d.loc[:, xVars].values
x = StandardScaler().fit_transform(x)
pca = PCA(n_components = 2)
X = pca.fit_transform(x)

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
               SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
               DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max_\
               _features=1), \
               AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42)
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name, clf.score(X_test, y_test))
    expected, predicted = y_test, clf.predict(X_test)
    print(metrics.classification_report(expected, predicted))
    print(metrics.confusion_matrix(expected, predicted))
    print('-' * 60)

```

```

0     819
1     136
Name: etiquetas, dtype: int64
Nearest Neighbors 0.818815331010453
      precision    recall   f1-score   support
          0       0.87      0.92      0.90      241
          1       0.41      0.28      0.33       46

   micro avg       0.82      0.82      0.82      287
   macro avg       0.64      0.60      0.61      287
weighted avg       0.80      0.82      0.81      287

```

```

[[222  19]
 [ 33  13]]
-----
```

```

Linear SVM 0.8397212543554007
      precision    recall   f1-score   support
```

0	0.84	1.00	0.91	241
1	0.00	0.00	0.00	46

micro avg	0.84	0.84	0.84	287
macro avg	0.42	0.50	0.46	287
weighted avg	0.71	0.84	0.77	287

```
[[241  0]
 [ 46  0]]
```

RBF SVM 0.8397212543554007

	precision	recall	f1-score	support
0	0.84	1.00	0.91	241
1	0.00	0.00	0.00	46

micro avg	0.84	0.84	0.84	287
macro avg	0.42	0.50	0.46	287
weighted avg	0.71	0.84	0.77	287

```
[[241  0]
 [ 46  0]]
```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
 warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning: Data with input dtype int64 was converted to float64 by StandardScaler.
 warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
 'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
 'precision', 'predicted', average, warn_for)

Gaussian Process 0.8397212543554007

	precision	recall	f1-score	support
0	0.84	1.00	0.91	241
1	0.00	0.00	0.00	46

micro avg	0.84	0.84	0.84	287
macro avg	0.42	0.50	0.46	287
weighted avg	0.71	0.84	0.77	287

```
[[241  0]
 [ 46  0]]
```

Decision Tree 0.8362369337979094

	precision	recall	f1-score	support
0	0.84	0.99	0.91	241
1	0.33	0.02	0.04	46

micro avg	0.84	0.84	0.84	287
macro avg	0.59	0.51	0.48	287
weighted avg	0.76	0.84	0.77	287

```
[[239  2]
 [ 45  1]]
```

Random Forest 0.8362369337979094

	precision	recall	f1-score	support
0	0.84	0.99	0.91	241

```

1      0.40    0.04    0.08     46
micro avg    0.84    0.84    0.84    287
macro avg    0.62    0.52    0.49    287
weighted avg  0.77    0.84    0.78    287
[[ 238   3]
 [ 44   2]]
-----
AdaBoost  0.8501742160278746
precision    recall   f1-score  support
0           0.85    1.00    0.92     241
1           1.00    0.07    0.12      46
micro avg    0.85    0.85    0.85    287
macro avg    0.92    0.53    0.52    287
weighted avg  0.87    0.85    0.79    287
[[ 241   0]
 [ 43   3]]
-----
Naive Bayes 0.8397212543554007
precision    recall   f1-score  support
0           0.84    1.00    0.91     241
1           0.00    0.00    0.00      46
micro avg    0.84    0.84    0.84    287
macro avg    0.42    0.50    0.46    287
weighted avg  0.71    0.84    0.77    287
[[ 241   0]
 [ 46   0]]
-----
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)

```

Estas matrices de confusión mejoran un poco, con una sola etiqueta aparecen valores más altos en la diagonal y menor valor de errores. La de mejor resultado es la Nearest Neighbors con 0.818815331010453, con 222 valores correctos contra 19 incorrectos, y 13 correctos contra 33 incorrectos. Se intenta agregar más atributos al modelo para realizar un ajuste de parámetros que pueda ayudar a mejorar la clasificación, se agregan datos nuevos de los internos: el crimen que cometieron, el nivel de educación, los años de sentencia y el género. Como etiqueta sigue el mismo atributo, los ingresos mensuales por encima de 9000 pesos.

```
In [136]: if hasattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context
o = pd.read_csv("practica6.csv")
d = pd.concat([o.loc[o.age >= 16], o.loc[~isnan(o.age)]])

etiqueta = d.incomePrev >= 4

cat = pd.Categorical(d.crimen)
d.crimen = cat.codes
cat2 = pd.Categorical(d.educacion)
d.educacion = cat2.codes
cat3 = pd.Categorical(d.gender)
d.gender = cat3.codes

xv = ['age', 'crimen', 'educacion', 'sentenceYears', 'gender']
```

```

d = d.loc[:, xv]

d['etiqueta'] = etiqueta
y = d.etiqueta
print(y.value_counts())
d = d.drop(['etiqueta'], 1)
X = StandardScaler().fit_transform(d.values)
names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
               SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
               DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max_\
               _features=1), \
               AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5)
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    y_m = clf.predict(X_test)
    print(name, metrics.classification_report(y_test, y_m))
    print(metrics.confusion_matrix(y_test, y_m))
    print('-' * 60)

False      819
True       136
Name: etiqueta, dtype: int64
Nearest Neighbors          precision      recall   f1-score   support
              False        0.89        0.96        0.92      414
              True        0.47        0.22        0.30       64
              micro avg     0.86        0.86        0.86      478
              macro avg     0.68        0.59        0.61      478
              weighted avg  0.83        0.86        0.84      478

[[398  16]
 [ 50 14]]
-----
Linear SVM          precision      recall   f1-score   support
              False        0.87        1.00        0.93      414
              True        0.00        0.00        0.00       64
              micro avg     0.87        0.87        0.87      478
              macro avg     0.43        0.50        0.46      478
              weighted avg  0.75        0.87        0.80      478

[[414   0]
 [ 64  0]]
-----
RBF SVM           precision      recall   f1-score   support
              False        0.89        0.99        0.94      414
              True        0.73        0.17        0.28       64
              micro avg     0.88        0.88        0.88      478
              macro avg     0.81        0.58        0.61      478
              weighted avg  0.87        0.88        0.85      478

[[410   4]
 [ 53 11]]
-----
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:
  Data with input dtype int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595: DataConversionWarning:
  Data with input dtype int64 was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning:

```

```

-----[elided]-----
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
    'precision', 'predicted', average, warn_for)

Gaussian Process
precision      recall   f1-score   support
  False        0.87      1.00      0.93     414
  True         0.00      0.00      0.00      64

  micro avg    0.87      0.87      0.87     478
  macro avg    0.43      0.50      0.46     478
weighted avg   0.75      0.87      0.80     478

[[414  0]
 [ 64  0]]]

Decision Tree
precision      recall   f1-score   support
  False        0.89      0.98      0.93     414
  True         0.57      0.19      0.28      64

  micro avg    0.87      0.87      0.87     478
  macro avg    0.73      0.58      0.61     478
weighted avg   0.84      0.87      0.84     478

[[405  9]
 [ 52 12]]]

Random Forest
precision      recall   f1-score   support
  False        0.88      0.99      0.93     414
  True         0.54      0.11      0.18      64

  micro avg    0.87      0.87      0.87     478
  macro avg    0.71      0.55      0.56     478
weighted avg   0.83      0.87      0.83     478

[[408  6]
 [ 57  7]]]

AdaBoost
precision      recall   f1-score   support
  False        0.88      0.95      0.92     414
  True         0.35      0.17      0.23      64

  micro avg    0.85      0.85      0.85     478
  macro avg    0.62      0.56      0.57     478
weighted avg   0.81      0.85      0.82     478

[[394 20]
 [ 53 11]]]

Naive Bayes
precision      recall   f1-score   support
  False        0.86      0.96      0.91     414
  True         0.06      0.02      0.03      64

  micro avg    0.84      0.84      0.84     478
  macro avg    0.46      0.49      0.47     478
weighted avg   0.76      0.84      0.79     478

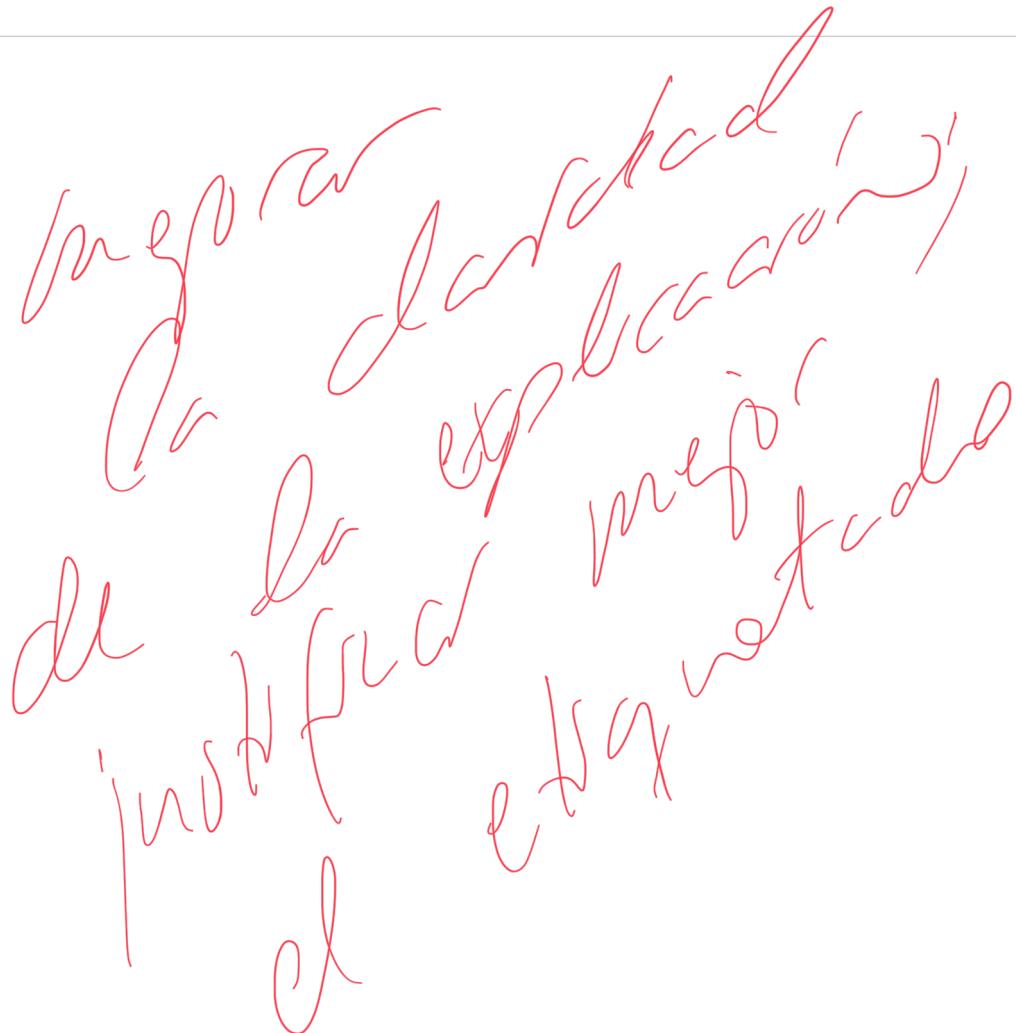
[[399 15]
 [ 63  1]]]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples.
    'precision', 'predicted', average, warn_for)

```

En las matrices anteriores algunos clasificadores llegaron a dar valor 0 en los erróneos, pero este ajuste de parámetros no logró perfección en los clasificadores. Esto indica que a pesar de que en prácticas anteriores el ingreso mensual de los internos fue un aspecto significativo en el tipo de delito a cometer, en este caso no resulta un elemento significativo para caracterizar a los internos.

In []:



1. Discusión de la Práctica 13

La **Práctica 13** fue basada en el Análisis de imágenes. Esta práctica tiene como objetivo extraer la información posible que se encuentra en formato imagen referente al caso de estudio de los internos. Las imágenes que se utilizan para realizar el trabajo no están en una base de datos, estas se encuentran en libros que han sido redactados a partir de información obtenida de las encuestas realizadas a los internos. En estos libros se encuentran imágenes de las pinturas echas por los internos durante su periodo de privación de la libertad.

La tarea no pudo ser entregada en tiempo ya que coincidió con tareas a entregar de la tesis, se entregó la tarea unas horas mas tarde de las que se debió entregar.

REPORTE PRÁCTICA 13: Análisis de imágenes

Análisis de Datos Multivariado

Caso de Estudio:

Condiciones en los Centros de Reinserción Social CERESO y Topo Chico

Evely Gutiérrez Noda #1935050

Introducción

En el siguiente reporte se analiza el caso de estudio de los internos del CERESO "Apodaca" y el Centro Preventivo de Reinserción Topo Chico. Los datos que se usan para este estudio provienen de encuestas realizadas a mujeres y hombres, que se encuentran privados de libertad dentro del Centro de Reinserción.

Esta práctica tiene como objetivo extraer la información posible que se encuentra en formato imagen referente al caso de estudio de los internos. Las imágenes que se utilizan para realizar el trabajo no están en una base de datos, estas se encuentran en libros que han sido redactados a partir de información obtenida de las encuestas realizadas a los internos. En estos libros se encuentran imágenes de las pinturas echas por los internos durante su periodo de privación de la libertad.

Para comenzar el trabajo se necesita instalar las librerías OpenCV y Pillow desde la consola, ya que estas cuentan con funcionalidades para el análisis de imágenes permitiendo acceder a la información que puedan tener. Estas librerías se instalan con las siguientes líneas en consola:

```
pip3 install opencv-python pip3 install Pillow
```

Como las imágenes que se van a utilizar están en un libro físico, se toman fotografías a estas y se guardan en PNG ya que es un formato que permite el acceso a los pixeles. Se utilizan 3 dibujos de los internos con los nombres "Mi Sueño", "Mundo Feliz", "Descubre la luz al salir".

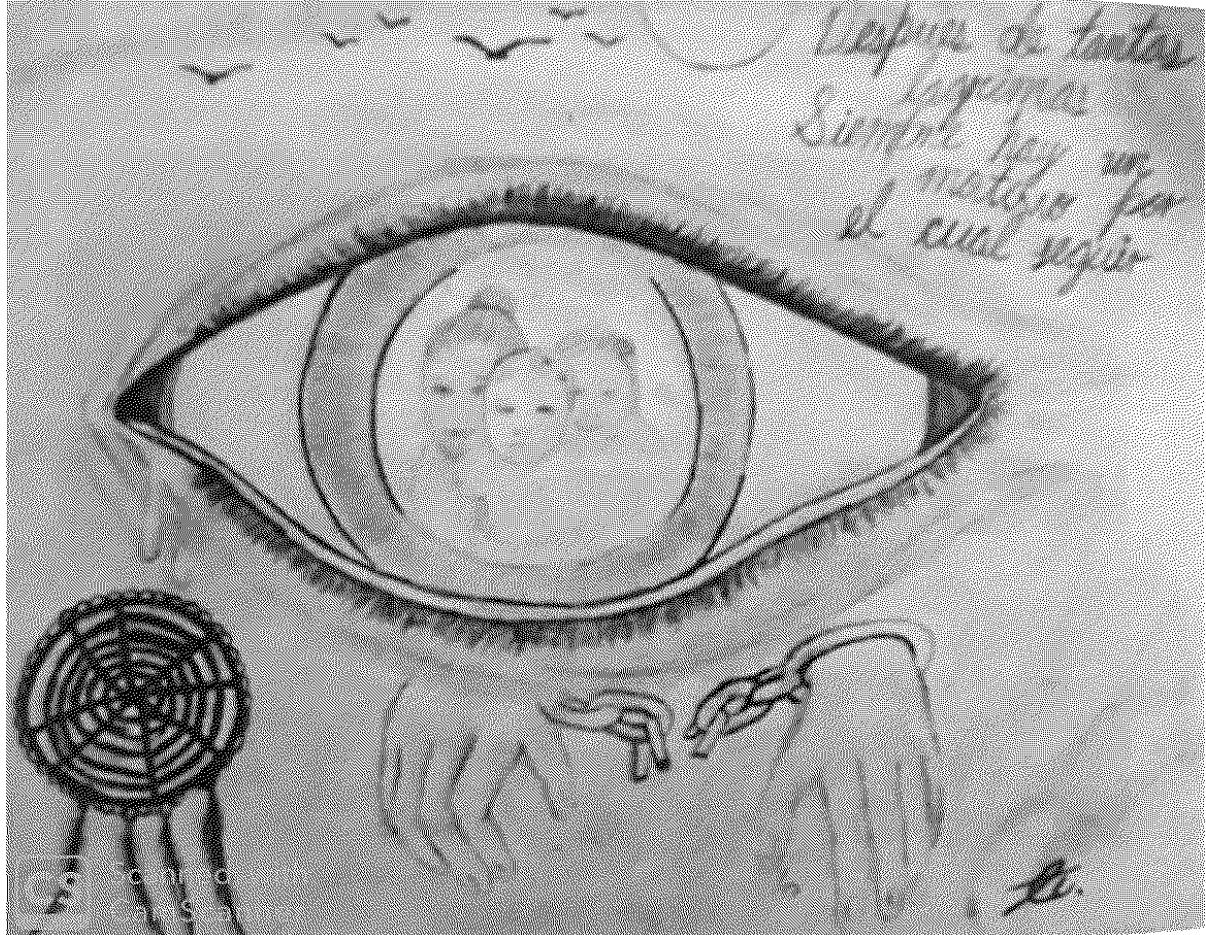
Se intenta interpretar los dibujos de los internos ya que de algún modo podrían influir en su estado de ánimo, del modo que, a mayor cantidad de dibujo en la hoja, pudiera ser mayor frustración o depresión del interno. Basándose en esta hipótesis podría ser que, a mayor cantidad de pixeles negros en la imagen, mayor cantidad de dibujo y por consecuente mayor frustración o depresión.

Para empezar, se elimina el color de fondo para el caso de que las hojas hayan sido de color, luego se convierte todo el texto a un mismo color para el caso de que utilicen varios colores de tinta al dibujar. Este proceso se conoce como binarización de imagen y se logra en este caso en Python con la librería Pillow .

```
In [67]: from PIL import Image  
imagen = Image.open("imagen1.png")  
nuevo = imagen.convert('1')  
print(nuevo.size)  
nuevo
```

(1162, 914)

Out[67]:



Ya queda la imagen en blanco y negro, pero tiene muchos pixeles aislados que pudieran no pertenecer al dibujo. Se intenta eliminar los pixeles aislados que no tienen ninguno adyacente.

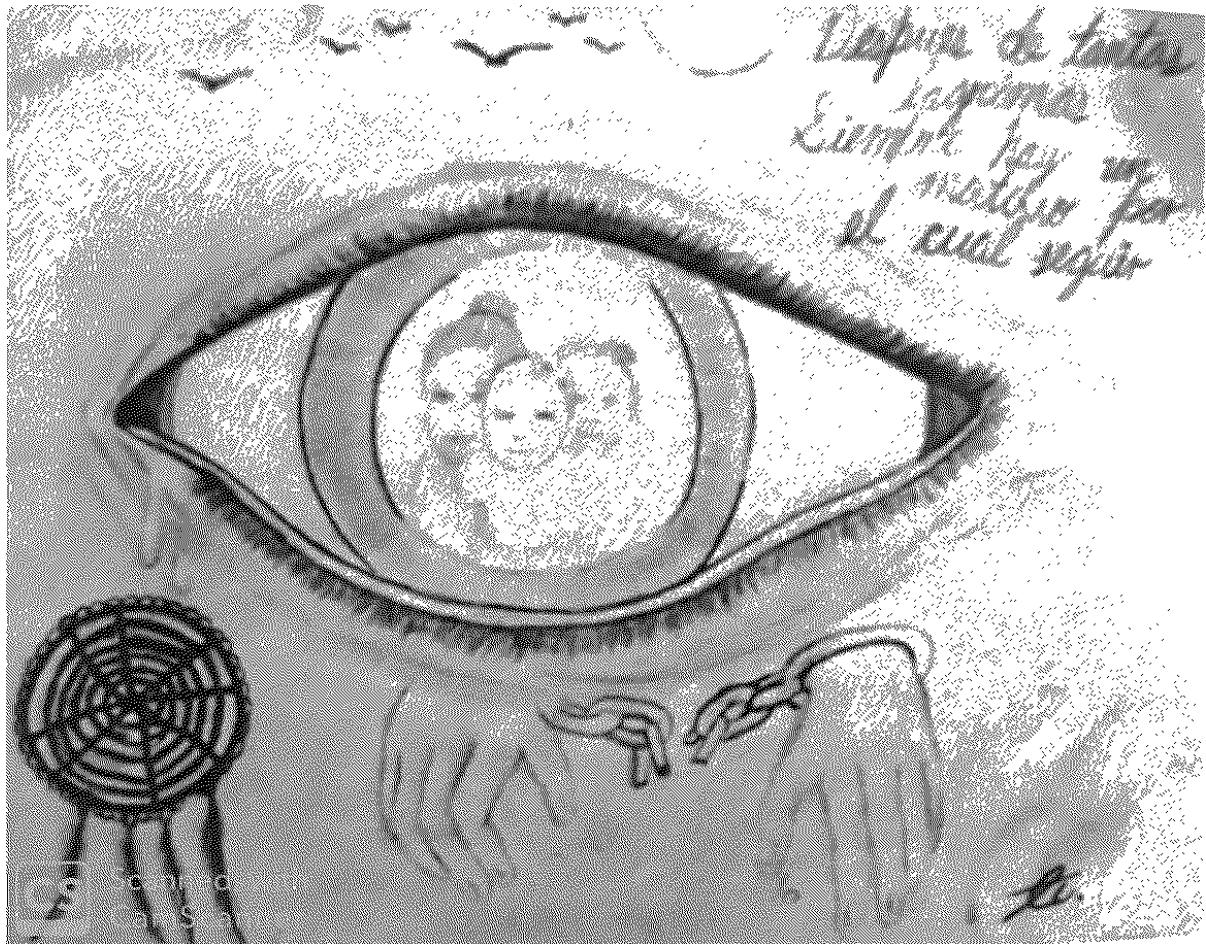
```
In [21]: import ssl
import requests
from PIL import Image

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

imagen = Image.open("Imagen1.jpg")
nuevo = imagen.convert('1')
P = nuevo.load()
ancho, altura = nuevo.size
borrados = 0
vecinos = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
for fila in range(altura):
    for columna in range(ancho):
        if P[columna, fila] == 0: # pixel es negro
            tiene = False
            for (df, dc) in vecinos:
                vf = fila + df
                vc = columna + dc
                if vf >= 0 and vc >= 0 and vf < altura and vc < ancho: # si existe el vecino
                    if P[vc, vf] == 0:
                        tiene = True
                        break # no hace falta seguir examinando
            if not tiene: # no tenía vecino negro
                P[columna, fila] = 255 # será blanco
                borrados += 1
print(borrados, "pixeles negros aislados eliminados")
nuevo
```

76635 pixeles negros aislados eliminados

Out[21]:



Se eliminan 76635 pixeles negros y se nota alguna mejoría en algunas partes de la imagen, sobre todo a la derecha, pero aún quedan muchas zonas con pixeles negros aislados, queda mucho ruido en el fondo aún. Se intenta quitar los pixeles que solo tienen un vecino negro para lograr menos ruido en el fondo.

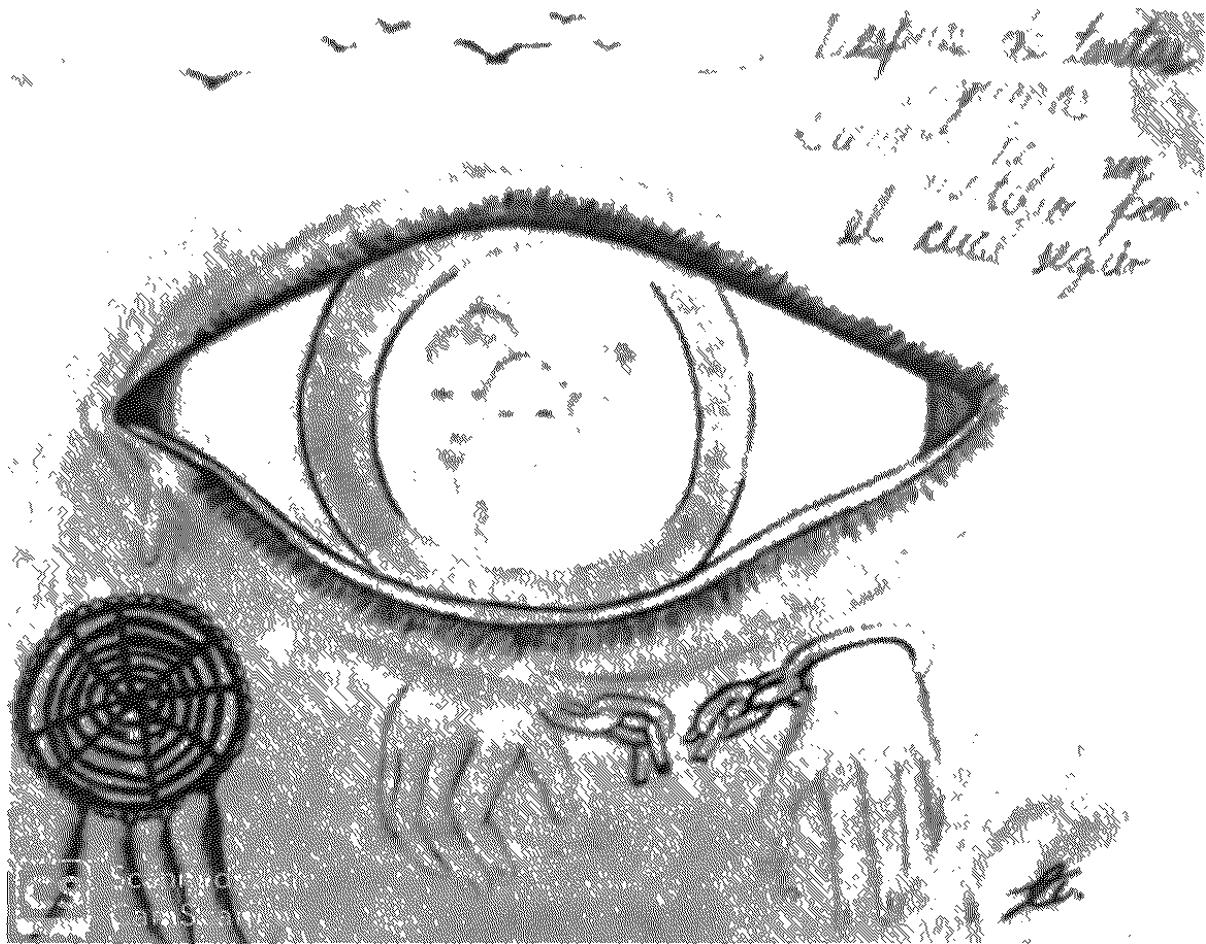
```
In [22]: import ssl
import requests
from PIL import Image

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

imagen = Image.open("imagen1.jpg")
nuevo = imagen.convert('1')
P = nuevo.load()
ancho, altura = nuevo.size
borrados = 0
vecinos = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
for fila in range(altura):
    for columna in range(ancho):
        if P[columna, fila] == 0: # pixel es negro
            contador = 0
            for (df, dc) in vecinos:
                vf = fila + df
                vc = columna + dc
                if vf >= 0 and vc >= 0 and vf < altura and vc < ancho: # si existe el vecino
                    if P[vc, vf] == 0:
                        contador += 1
            if contador < 2: # uno o cero vecinos negros
                P[columna, fila] = 255 # será blanco
                borrados += 1
print(borrados, "pixeles negros eliminados")
nuevo
```

156117 pixeles negros eliminados

Out[22]:



Continúa mejorando ya que se eliminaron 156117 pixeles, pero aun la parte izquierda de la imagen queda con mucho ruido, esto pudiera ser a causa de alguna sombra al tomar la fotografía del libro. Se realiza un conteo de pixeles para conocer el porcentaje de estos que quedaron negros para conocer cuento dibujo el interno en esa imagen.

```
In [58]: import ssl
import requests
from PIL import Image, ImageDraw

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

imagen = Image.open("Imagen1.jpg")
n = imagen.convert('1')
w, h = n.size
rgb = n.convert('RGB')
P = rgb.load()
negro = (0, 0, 0)
blanco = (255, 255, 255)
for f in range(h): # bordes verticales
    if P[0, f] == negro:
        ImageDraw.floodfill(rgb, (0, f), blanco)
    if P[w - 1, f] == negro:
        ImageDraw.floodfill(rgb, (w - 1, f), blanco)
for c in range(w): # bordes horizontales
    if P[c, 0] == negro:
        ImageDraw.floodfill(rgb, (c, 0), blanco)
    if P[c, h - 1] == negro:
        ImageDraw.floodfill(rgb, (c, h - 1), blanco)
n = rgb.convert('1')
P = n.load()
V = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
negros = 0
totales = w * h
for f in range(1, h - 1): # sin bordes ahora
    for c in range(1, w - 1):
        if P[c, f] == 0:
            cont = 0
            for (df, dc) in V:
                if P[c + dc, f + df] == 0: # siempre existen
                    cont += 1
            if cont < 2:
                P[c, f] = 255 # blanco
            else:
                negros += 1
print('{:.2f} porciento negro'.format(100 * negros / totales))
```

18.06 porciento negro

Se compara con las otras dos imágenes.

```
In [ ]: E:\Maestria\4rto semestre\Analisis Estadistico Multivariado>curl -o imagen2.png https://github.com/EvelyGutierrez/Analisis-Estadistico-Multivariado/blob/master/imagen2.png
          % Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100  63075     0  63075     0      0   116k      0 --::-- --::-- --::--  116k

E:\Maestria\4rto semestre\Analisis Estadistico Multivariado>curl -o imagen3.png https://github.com/EvelyGutierrez/Analisis-Estadistico-Multivariado/blob/master/imagen3.png
          % Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100  63075     0  63075     0      0  73513      0 --::-- --::-- --::--  73513
```

```
In [30]: import ssl
import requests
from PIL import Image, ImageDraw

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

for ejemplo in range(1, 4): # de uno a cinco
    #url = 'https://github.com/EvelyGutierrez/Analisis-Estadistico-Multivariado/imagen{:d}.png'.format(ejemplo)
    if ejemplo == 1:
        imagen = Image.open("imagen1.png")
    if ejemplo == 2:
        imagen = Image.open("imagen2.png")
    if ejemplo == 3:
        imagen = Image.open("imagen3.png")
n = imagen.convert('1')
w, h = n.size
rgb = n.convert('RGB')
P = rgb.load()
negro = (0, 0, 0)
blanco = (255, 255, 255)
for f in range(h): # bordes verticales
    if P[0, f] == negro:
        ImageDraw.floodfill(rgb, (0, f), blanco)
    if P[w - 1, f] == negro:
        ImageDraw.floodfill(rgb, (w - 1, f), blanco)
for c in range(w): # bordes horizontales
    if P[c, 0] == negro:
        ImageDraw.floodfill(rgb, (c, 0), blanco)
    if P[c, h - 1] == negro:
        ImageDraw.floodfill(rgb, (c, h - 1), blanco)
n = rgb.convert('1')
P = n.load()
V = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
negros = 0
for f in range(1, h - 1): # sin bordes ahora
    for c in range(1, w - 1):
        if P[c, f] == 0:
            cont = 0
            for (df, dc) in V:
                if P[c + dc, f + df] == 0: # siempre existen
                    cont += 1
            if cont < 2:
                P[c, f] = 255 # blanco
            else:
                negros += 1

print('Ejemplo {:d}:\t{:d} pixeles de texto'.format(ejemplo, negros))
```

Ejemplo 1: 191682 pixeles de texto
 Ejemplo 2: 155414 pixeles de texto
 Ejemplo 3: 209351 pixeles de texto

Luego se revisa como quedaron las imágenes binarizadas utilizando el mismo procedimiento anterior de borrado de píxeles en las 3 imágenes.

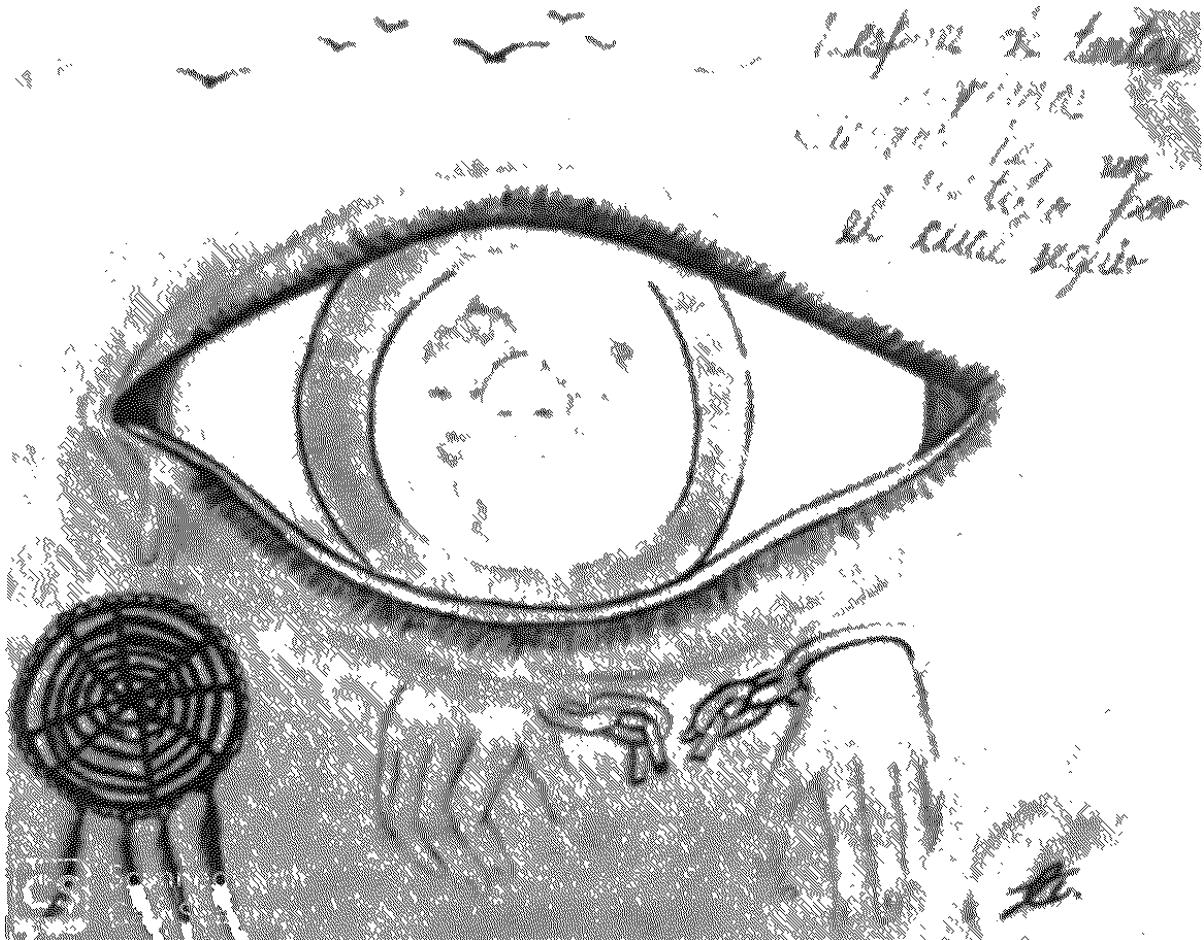
```
In [45]: import ssl
import requests
from PIL import Image, ImageDraw

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

for ejemplo in range(1, 4): # de uno a cinco
    if ejemplo == 1:
        imagen = Image.open("imagen1.png")
    if ejemplo == 2:
        imagen = Image.open("imagen2.png")
    if ejemplo == 3:
        imagen = Image.open("imagen3.png")
n = imagen.convert('1')
w, h = n.size
rgb = n.convert('RGB')
P = rgb.load()
negro = (0, 0, 0)
blanco = (255, 255, 255)
for f in range(h): # bordes verticales
    if P[0, f] == negro:
        ImageDraw.floodfill(rgb, (0, f), blanco)
    if P[w - 1, f] == negro:
        ImageDraw.floodfill(rgb, (w - 1, f), blanco)
for c in range(w): # bordes horizontales
    if P[c, 0] == negro:
        ImageDraw.floodfill(rgb, (c, 0), blanco)
    if P[c, h - 1] == negro:
        ImageDraw.floodfill(rgb, (c, h - 1), blanco)
n = rgb.convert('1')
P = n.load()
V = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
for f in range(1, h - 1): # sin bordes ahora
    for c in range(1, w - 1):
        if P[c, f] == 0:
            cont = 0
            for (df, dc) in V:
                if P[c + dc, f + df] == 0: # siempre existen
                    cont += 1
            if cont < 2:
                P[c, f] = 255 # blanco
n.save('bin_{:d}.png'.format(ejemplo))

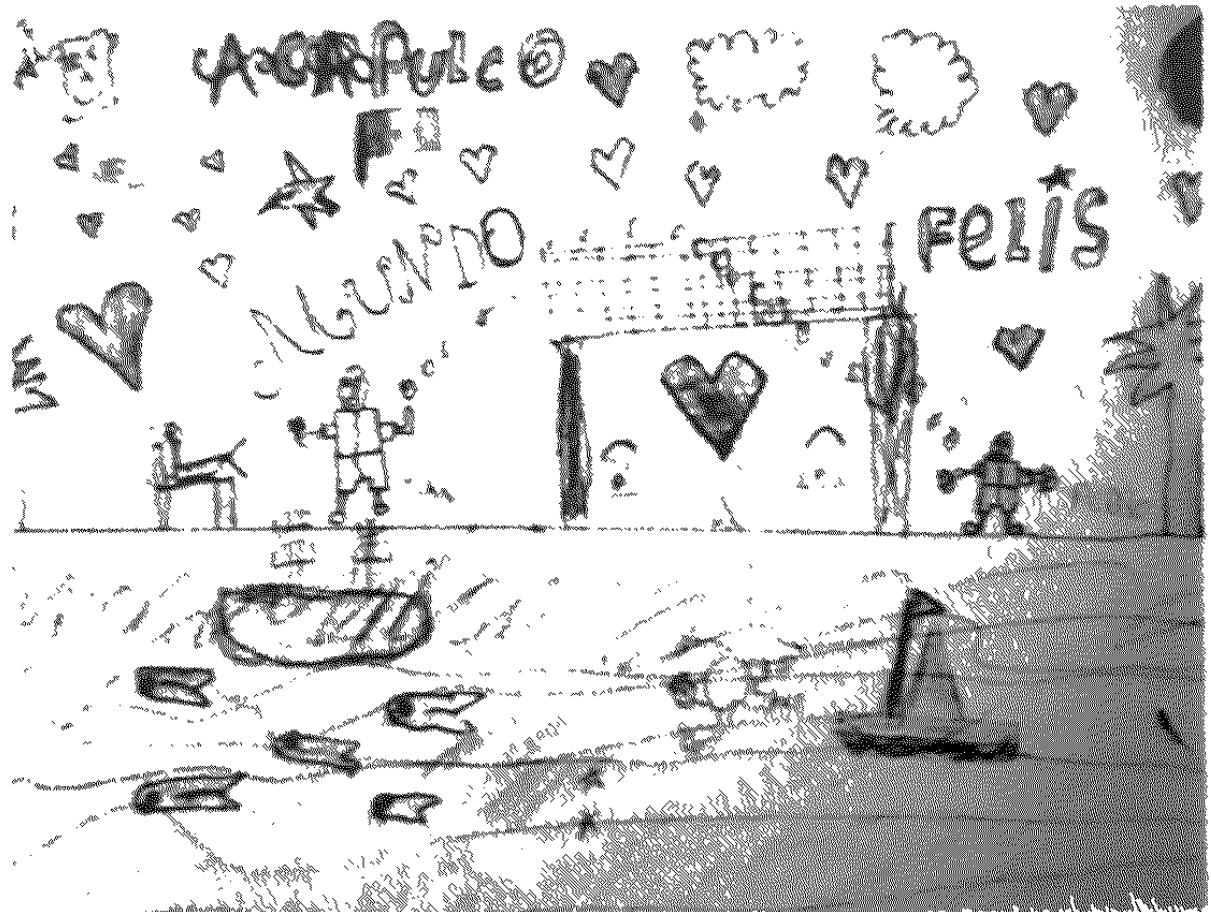
imagen1 = Image.open("bin_1.png")
imagen1
```

Out[45]:



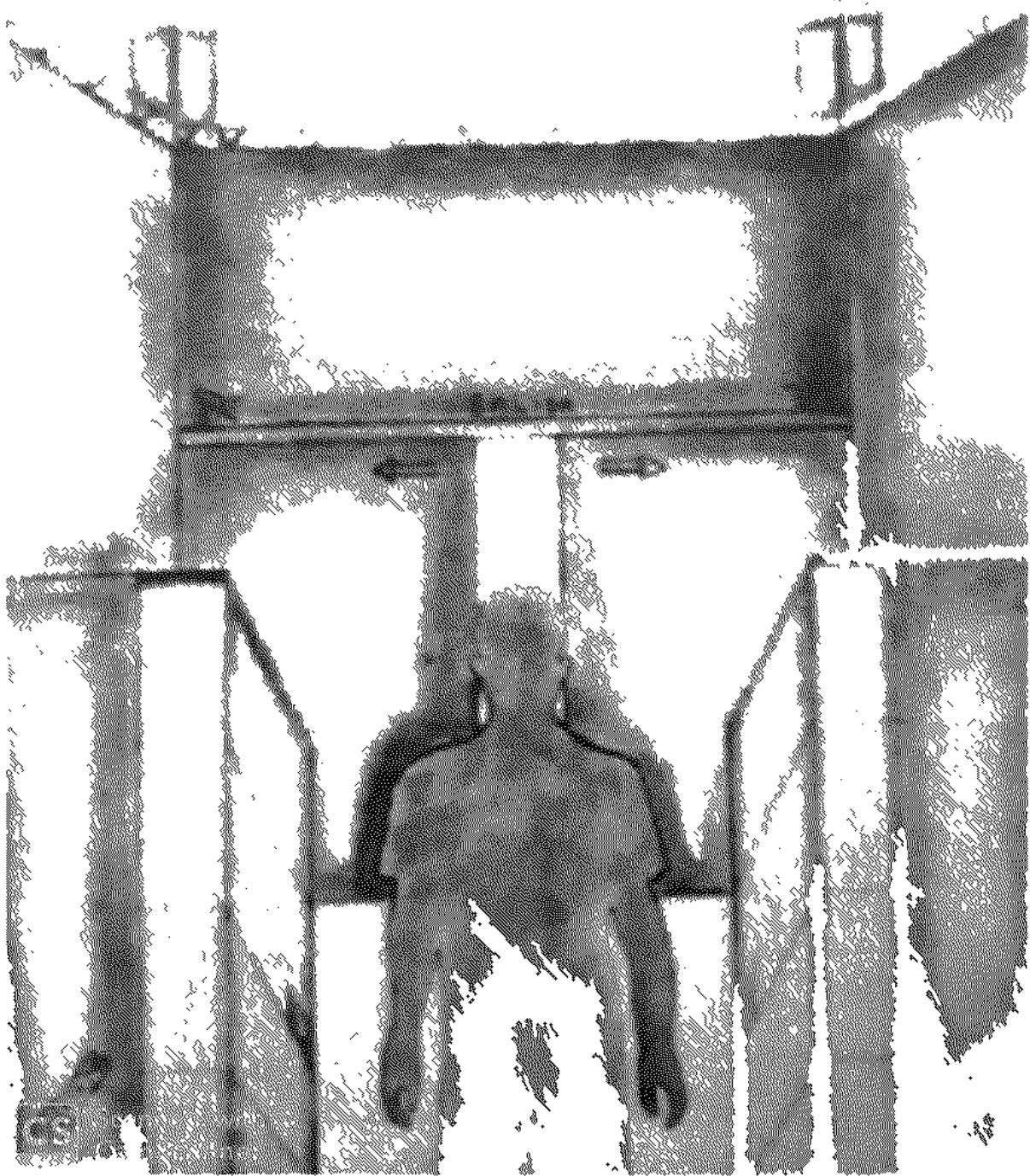
```
In [43]: imagen2 = Image.open("bin_2.png")
imagen2
```

Out[43]:



```
In [44]: imagen3 = Image.open("bin_3.png")
imagen3
```

Out[44]:



Funciona bastante parecido en las 1ras dos imágenes, pero como los dibujos de los internos generalmente tienen sombras echas con el lápiz afecta mucho este proceso dejando mucho ruido en el fondo, además de que son copias de los dibujos, no los originales.

Como la binarización no funciona muy bien en este tipo de imágenes, se prueba con otro proceso, convirtiendo en blanco todos los pixeles que no sean negros y se binariza levemente. Esto se logra convirtiendo la imagen a RGB.

```
In [46]: import ssl
import requests
from PIL import Image, ImageDraw

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

negro = (0, 0, 0)
blanco = (255, 255, 255)
umbral = 90 # tomó un poco de búsqueda binaria para hallarle un valor que funcione

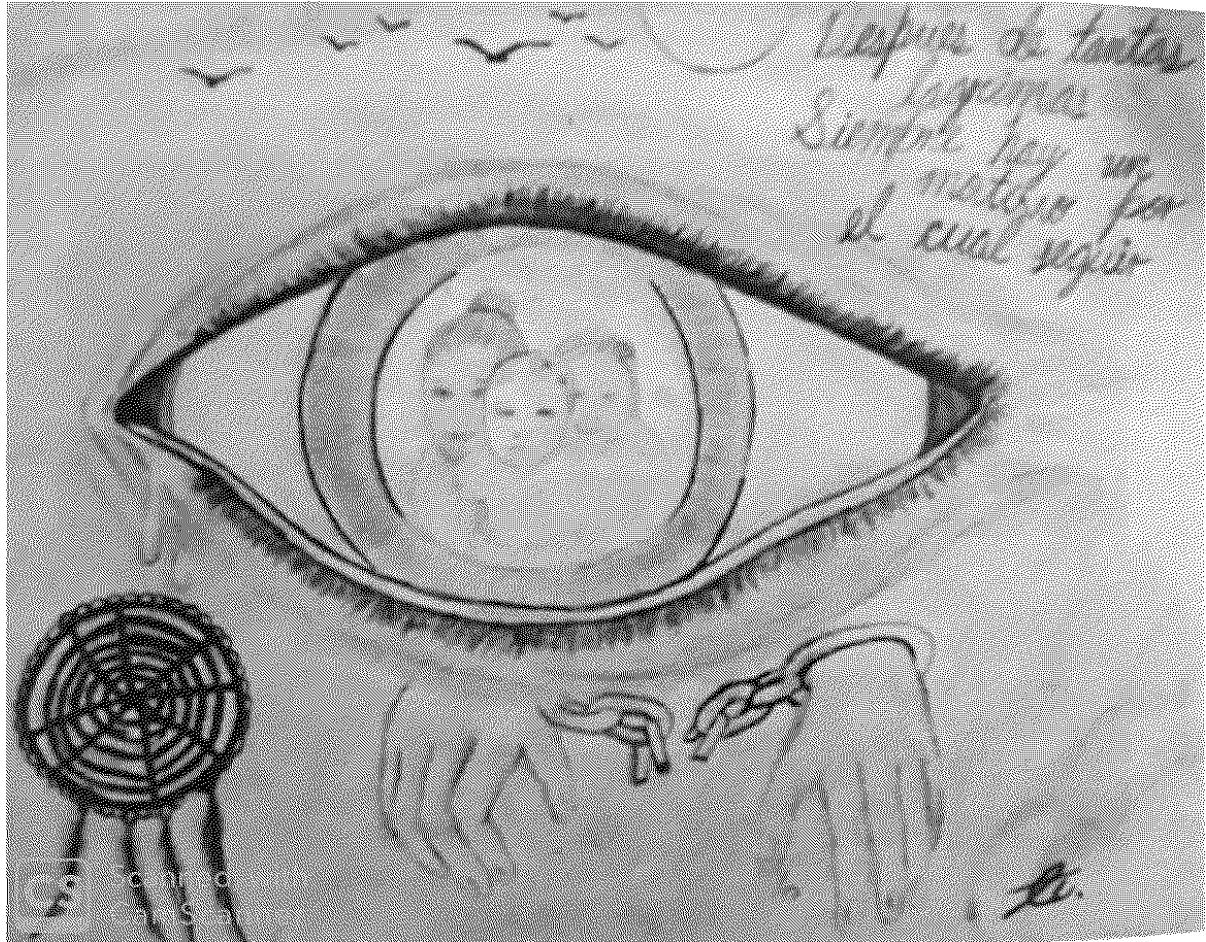
for ejemplo in range(1, 4): # de uno a cinco

    if ejemplo == 1:
        imagen = Image.open("imagen1.png")
    if ejemplo == 2:
        imagen = Image.open("imagen2.png")
    if ejemplo == 3:
        imagen = Image.open("imagen3.png")

    n = imagen.convert('RGB')
    w, h = n.size
    P = n.load()
    for f in range(h): # bordes verticales
        if P[0, f] == negro:
            ImageDraw.floodfill(n, (0, f), blanco)
        if P[w - 1, f] == negro:
            ImageDraw.floodfill(n, (w - 1, f), blanco)
    for c in range(w): # bordes horizontales
        if P[c, 0] == negro:
            ImageDraw.floodfill(n, (c, 0), blanco)
        if P[c, h - 1] == negro:
            ImageDraw.floodfill(n, (c, h - 1), blanco)
    for f in range(h):
        for c in range(w):
            rgb = P[c, f]
            if max(rgb) - min(rgb) > umbral: # tiene un color que no es gris
                P[c, f] = blanco
    b = n.convert('1') # binaricemos lo que queda
    b.save('rgb_{:d}.png'.format(ejemplo))
```

```
In [47]: imagen1 = Image.open("rgb_1.png")
imagen1
```

Out[47]:



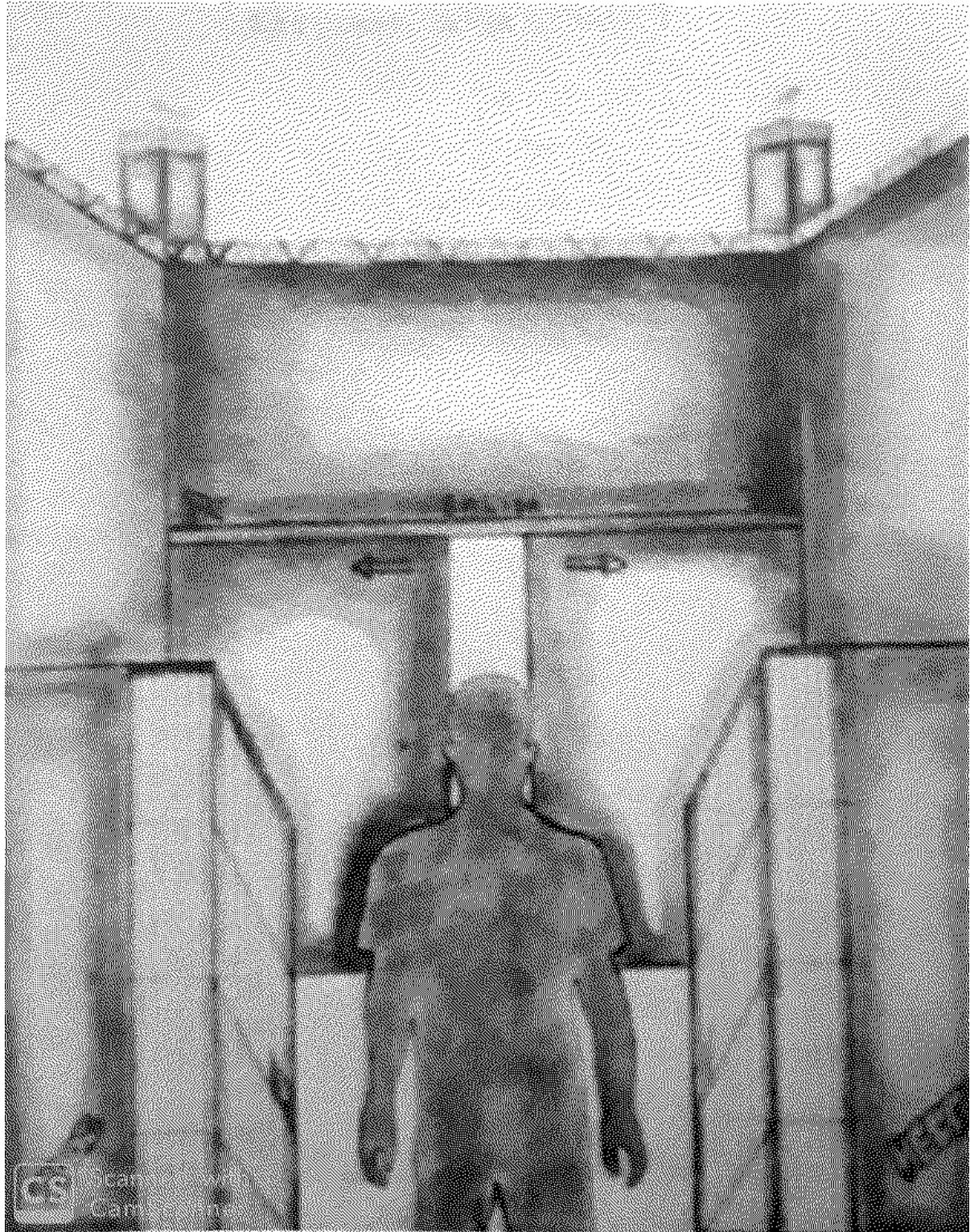
```
In [48]: imagen1 = Image.open("rgb_2.png")
imagen1
```

Out[48]:



```
In [49]: imagen1 = Image.open("rgb_3.png")
imagen1
```

Out[49]:



El cambio no es mucho con respecto a las imágenes originales, al parecer fue mejor el proceso de borrar pixeles negros aislados, ya que se logró algunos espacios completos en blanco, en cambio con el proceso de forzar a blanco los pixeles no negros, se logra solamente aclarar un poco las imágenes. De igual modo se realiza nuevamente un conteo de pixeles negros para ver como quedaron con este proceso convirtiendo la imagen a RGB.

```
In [50]: import ssl
import requests
from PIL import Image, ImageDraw

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

negro = (0, 0, 0)
blanco = (255, 255, 255)
umbral = 90

for ejemplo in range(1, 4): # de uno a cinco
    if ejemplo == 1:
        imagen = Image.open("imagen1.png")
    if ejemplo == 2:
        imagen = Image.open("imagen2.png")
    if ejemplo == 3:
        imagen = Image.open("imagen3.png")
    n = imagen.convert('RGB')
    w, h = n.size
    P = n.load()
    for f in range(h): # bordes verticales
        if P[0, f] == negro:
            ImageDraw.floodfill(n, (0, f), blanco)
        if P[w - 1, f] == negro:
            ImageDraw.floodfill(n, (w - 1, f), blanco)
    for c in range(w): # bordes horizontales
        if P[c, 0] == negro:
            ImageDraw.floodfill(n, (c, 0), blanco)
        if P[c, h - 1] == negro:
            ImageDraw.floodfill(n, (c, h - 1), blanco)
    for f in range(h):
        for c in range(w):
            rgb = P[c, f]
            if max(rgb) - min(rgb) > umbral: # tiene un color que no es gris
                P[c, f] = blanco
    b = n.convert('1') # binaricemos lo que queda
    P = b.load()
    negros = 0
    for f in range(h):
        for c in range(w):
            if P[c, f] == 0:
                negros += 1
    print(ejemplo, negros)
```

```
1 349550
2 275409
3 331287
```

Se verifican si la información obtenida hasta ahora correlaciona con el tamaño de la imagen original en PNG.

```
In [ ]: ls -la imagen?.png | awk '{print $5}'  
1290109  
1534390  
1252406  
  
ls -la rgb_*.png | awk '{print $5}'  
126016  
100412  
105205  
  
ls -la bin_*.png | awk '{print $5}'  
69948  
57164  
64450
```

```
In [62]: import ssl
import requests
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw
from pandas.plotting import scatter_matrix

data = pd.DataFrame({'PNG': [1290109, 1534390, 1252406], \
                     'bin': [69948, 57164, 64450]})

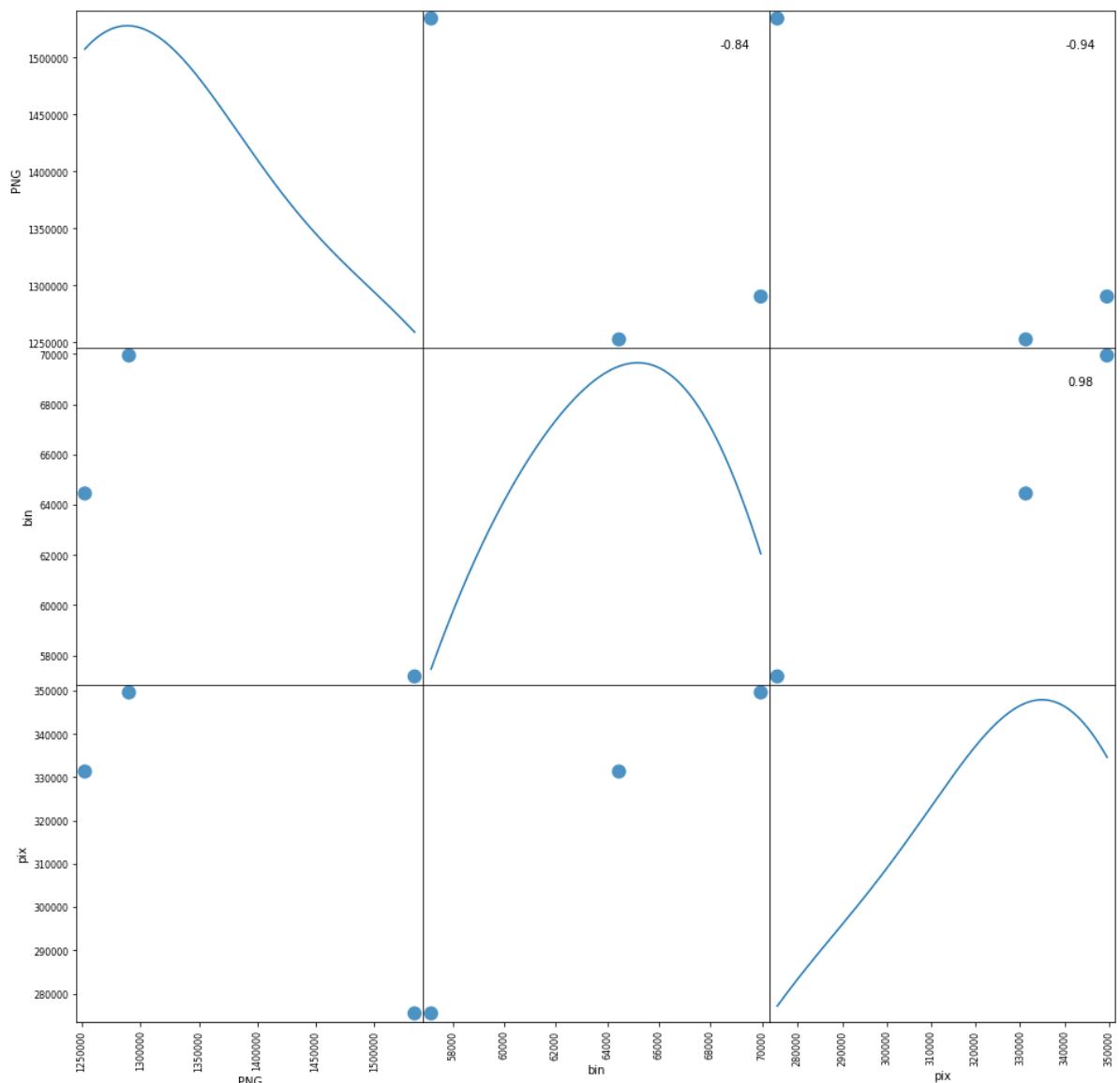
if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

negro = (0, 0, 0)
blanco = (255, 255, 255)
umbral = 90

pix = []
for ejemplo in range(1, 4): # de uno a cinco
    if ejemplo == 1:
        imagen = Image.open("imagen1.png")
    if ejemplo == 2:
        imagen = Image.open("imagen2.png")
    if ejemplo == 3:
        imagen = Image.open("imagen3.png")
    n = imagen.convert('RGB')
    w, h = n.size
    P = n.load()
    for f in range(h): # bordes verticales
        if P[0, f] == negro:
            ImageDraw.floodfill(n, (0, f), blanco)
        if P[w - 1, f] == negro:
            ImageDraw.floodfill(n, (w - 1, f), blanco)
    for c in range(w): # bordes horizontales
        if P[c, 0] == negro:
            ImageDraw.floodfill(n, (c, 0), blanco)
        if P[c, h - 1] == negro:
            ImageDraw.floodfill(n, (c, h - 1), blanco)
    for f in range(h):
        for c in range(w):
```

```
rgb = P[c, f]
    if max(rgb) - min(rgb) > umbral: # tiene un color que no es gris
        P[c, f] = blanco
b = n.convert('1') # binaricemos lo que queda
P = b.load()
negros = 0
for f in range(h):
    for c in range(w):
        if P[c, f] == 0:
            negros += 1
pix.append(negros)

data['pix'] = pix
# visualización basado en la discusión de
# https://stackoverflow.com/questions/27768677/pandas-scatter-matrix-display-correlation-coefficient
ax = scatter_matrix(data, alpha = 0.8, figsize = (16, 16), diagonal = 'kde', s = 600)
c = data.corr().values
for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
    ax[i, j].annotate('{:.2f}'.format(c[i, j]), (0.9, 0.9), \
                      xycoords = 'axes fraction', ha = 'center', va =
'center')
plt.show()
```



```
In [65]: import ssl
import requests
import numpy as np
import pandas as pd
import seaborn as sns
from PIL import Image, ImageDraw

data = pd.DataFrame({'lbl': [x for x in range(1, 4)], \
                     'PNG': [1290109, 1534390, 1252406], \
                     'bin': [69948, 57164, 64450]})

if getattr(ssl, '_create_unverified_context', None):
    ssl._create_default_https_context = ssl._create_unverified_context

negro = (0, 0, 0)
blanco = (255, 255, 255)
umbral = 90

pix = []
for ejemplo in range(1, 4): # de uno a cinco
    if ejemplo == 1:
```

```

        imagen = Image.open("imagen1.png")
if ejemplo == 2:
    imagen = Image.open("imagen2.png")
if ejemplo == 3:
    imagen = Image.open("imagen3.png")
n = imagen.convert('RGB')
w, h = n.size
P = n.load()
for f in range(h): # bordes verticales
    if P[0, f] == negro:
        ImageDraw.floodfill(n, (0, f), blanco)
    if P[w - 1, f] == negro:
        ImageDraw.floodfill(n, (w - 1, f), blanco)
for c in range(w): # bordes horizontales
    if P[c, 0] == negro:
        ImageDraw.floodfill(n, (c, 0), blanco)
    if P[c, h - 1] == negro:
        ImageDraw.floodfill(n, (c, h - 1), blanco)
for f in range(h):
    for c in range(w):
        rgb = P[c, f]
        if max(rgb) - min(rgb) > umbral: # tiene un color que no es gris
            P[c, f] = blanco
b = n.convert('1') # binaricemos lo que queda
P = b.load()
negros = 0
for f in range(h):
    for c in range(w):
        if P[c, f] == 0:
            negros += 1
pix.append(negros)

data['pix'] = pix

# etiquetado de https://stackoverflow.com/questions/46027653/adding-labels-in-x-y-scatter-plot-with-seaborn
def label_point(x, y, xoffset, yoffset, val, ax):
    a = pd.concat({'x': x, 'y': y, 'val': val}, axis=1)
    for i, point in a.iterrows():
        ax.text(point['x'] + xoffset, point['y'] + yoffset, str(point['val']))

sns.lmplot(x = 'PNG', y = 'bin', data=data, fit_reg = True, height = 5, aspect = 1.4)
label_point(data.PNG, data.bin, 50000, 9000, data.lbl, plt.gca())

sns.lmplot(x = 'PNG', y = 'pix', data=data, fit_reg = True, height = 5, aspect = 1.4)
label_point(data.PNG, data.pix, 50000, 9000, data.lbl, plt.gca())

sns.lmplot(x = 'bin', y = 'pix', data=data, fit_reg = True, height = 5, aspect = 1.4)
label_point(data.bin, data.pix, 3000, 3000, data.lbl, plt.gca())

```

